UIO forside her

Title for master thesis

Sabba Ifzal

May 1, 2014

Tom side

Acknowledgements

Takke folk som har hjulpet til

Veiledere – Torbjørn, Lex, Geir Kjetil og Ksenia

Andre folk i 10 – Sveinung, Kai, Jon

Ruth for sensur

Ghada for samtaler

etc

Abstract

Kort oppsummert hele oppgaven

Why you do this

How you do this

The results

Conclusion

Preface

Hvem er oppgaven ment for? Andre bioinformatikkstudenter

Contents

[Acknowledgements 5](#_Toc383649176)

[Abstract 6](#_Toc383649177)

[Preface 8](#_Toc383649178)

[Contents 10](#_Toc383649179)

[List of figures 12](#_Toc383649180)

[List of tables 13](#_Toc383649181)

[Chapter 1 Introduction 14](#_Toc383649182)

[1.1 Background 14](#_Toc383649183)

[1.2 PROBLEM STATEMENT / AIMs 14](#_Toc383649184)

[1.3 Literature review 14](#_Toc383649185)

[1.4 Problem solution 14](#_Toc383649186)

[Chapter 2 Background 15](#_Toc383649187)

[2.1 Genome Assembly 15](#_Toc383649188)

[2.1.1 Sequencing 15](#_Toc383649189)

[2.1.2 Assembly 16](#_Toc383649190)

[2.1.3 Quality measures 17](#_Toc383649191)

[2.2 Bioinformatics 18](#_Toc383649192)

[2.2.1 Benchmarking 19](#_Toc383649193)

[2.3 Galaxy 19](#_Toc383649194)

[Chapter 3 Material 20](#_Toc383649195)

[3.1 Datasets 20](#_Toc383649196)

[3.1.1 GAGE-B 20](#_Toc383649197)

[3.1.2 Galaxy 21](#_Toc383649198)

[3.2 Reference genomes 22](#_Toc383649199)

[3.3 Software 22](#_Toc383649200)

[Chapter 4 Methods 24](#_Toc383649201)

[4.1 Reproducing the GAGE-B results 24](#_Toc383649202)

[4.2 Galaxy 24](#_Toc383649203)

[Chapter 5 Results 25](#_Toc383649204)

[Chapter 6 Discussion 26](#_Toc383649205)

[6.1 Interpreting the GAGE-B results 26](#_Toc383649206)

[6.2 Analysis of galaxy tool 26](#_Toc383649207)

[6.2.1 Performance 26](#_Toc383649208)

[6.2.2 Potential use 26](#_Toc383649209)

[6.2.3 Weakness 26](#_Toc383649210)

[6.3 Further work 26](#_Toc383649211)

[6.4 Conclusion 26](#_Toc383649212)

[Glossary 27](#_Toc383649213)

[Appendix a (Plot/Figure) 29](#_Toc383649214)

[Appendix B (Tables) 30](#_Toc383649215)

List of figures

[figure 1 29](#_Toc377132394)

[Table 7-1 Bla bla bla 30](#_Toc377132395)

List of tables

[figure 1 29](#_Toc377132379)

[Table 7-1 Bla bla bla 30](#_Toc377132380)

# Introduction

## Background

DNA sequencing technology such as Next Generation Sequencing (NGS) is developing and revolutionizing the field of sequencing, allowing scientists to determine the sequence of nucleotides with an extreme speed. The task of puzzling together small pieces of a sequence from a new genome into larger continuous parts is performed by software tools such as CELERA/CABOG [[1](#_ENREF_1)], Velvet [[2](#_ENREF_2)] and ABySS [[3](#_ENREF_3)] among others. This is a difficult task and is performed with many adjustable parameters and varying speed and results, making the assessment of the algorithms used by the software tools important. Even though some tools, such as QUAST [[4](#_ENREF_4)], that measure the quality of a certain method exist, the bioinformatics field lacks enough information about which method would perform best under certain conditions. There have been some attempts on assessment resulting in benchmarks such as GAGE[[5](#_ENREF_5)] and Assemblathon 1 and 2 [[6](#_ENREF_6), [7](#_ENREF_7)], but in general, the development of benchmarks is slower than the development of assembly methods in itself, making the needs for a new system even more urgent.

Some biologists have pointed out that one of the disadvantages of current benchmarks is the lack of enough visual images, such as bar charts and scatterplots. Others want a system that is technically advanced, but user-friendly so that less experienced computer-users can easily adapt to the use of the system. This can be performed by reducing the number of required installation, creating makefiles or by reducing the number of steps required to get an assessment of an assembly. With the rising numbers of new assemblers, each proclaiming to be better than the previous version or the competitor, the need for a system which can give a comparison of old against new with minimal effort is desirable. The desired outcome for a new system is something that will reduce the installation requirement and increase the assessment statistics with more visual parameters such as custom designed plots depending on the users need.

**Why decide to have this project…**

**Why should the project be done, why use time**

## Problem statement / Aims

This Galaxy tool will be used for the assessment of how well various tools for genome assembly, short sequence mapping or variant calling perform.

What you actually doing in the project.

Visualize quastresult and add information quast-like, that can be useful for determining which assembly is the best.

Same principle can be added.

Organize a automatized principle. Make something to compare.

## Literature review

Quast, gage assemblathon

Hvordan andre har funnet den beste løsningen

## Problem solution

**Briefly about the solution. How you solved it.**  
The solution to the increasing need of a new benchmark was to develop a system that combined the good statistical output from QUAST with the flexibility of custom code and visualization in the Galaxy framework. Since QUAST has a rather good output structure, the Galaxy tool reuses this structure with some modifications, using python and javascript, to give the users a more tailored view of the output that can be viewed, modified and rerun as input for the next assessment.

**Benefit of system. Quast need install etc,**   
Using the Galaxy framework to create a tool to compare assemblies benefits future user because they ***do not need to install anything*** as long as it is running on the University of Oslo’s server: *invitro.titan.uio.no:24688*. This is good news for those who get frustrated for having to download, compile, install and run everything separately. All the users need to do is create a user-account (if they want to store their results), upload their assemblies, or copy datasets that other users have published, and run the tool. If anyone wants the tool on their own server, then all they need to do is copy the tool folder from github-repository to their Galaxy instance and add proper link to the tool as described in the attached readme file.

One of the advantages of this tool compared to for instance QUAST is that if a new dataset or assembler is available, then the user can effortlessly compare an old Galaxy-result with the output from the new assembly. This can be done since Galaxy stores each run with its parameters as an element in current history (Figure 1-1). The user save time because they only need to add the old result as one parameter and the new assembly as the second parameter instead of manually adding all the old datasets, the new assembly and other parameters.

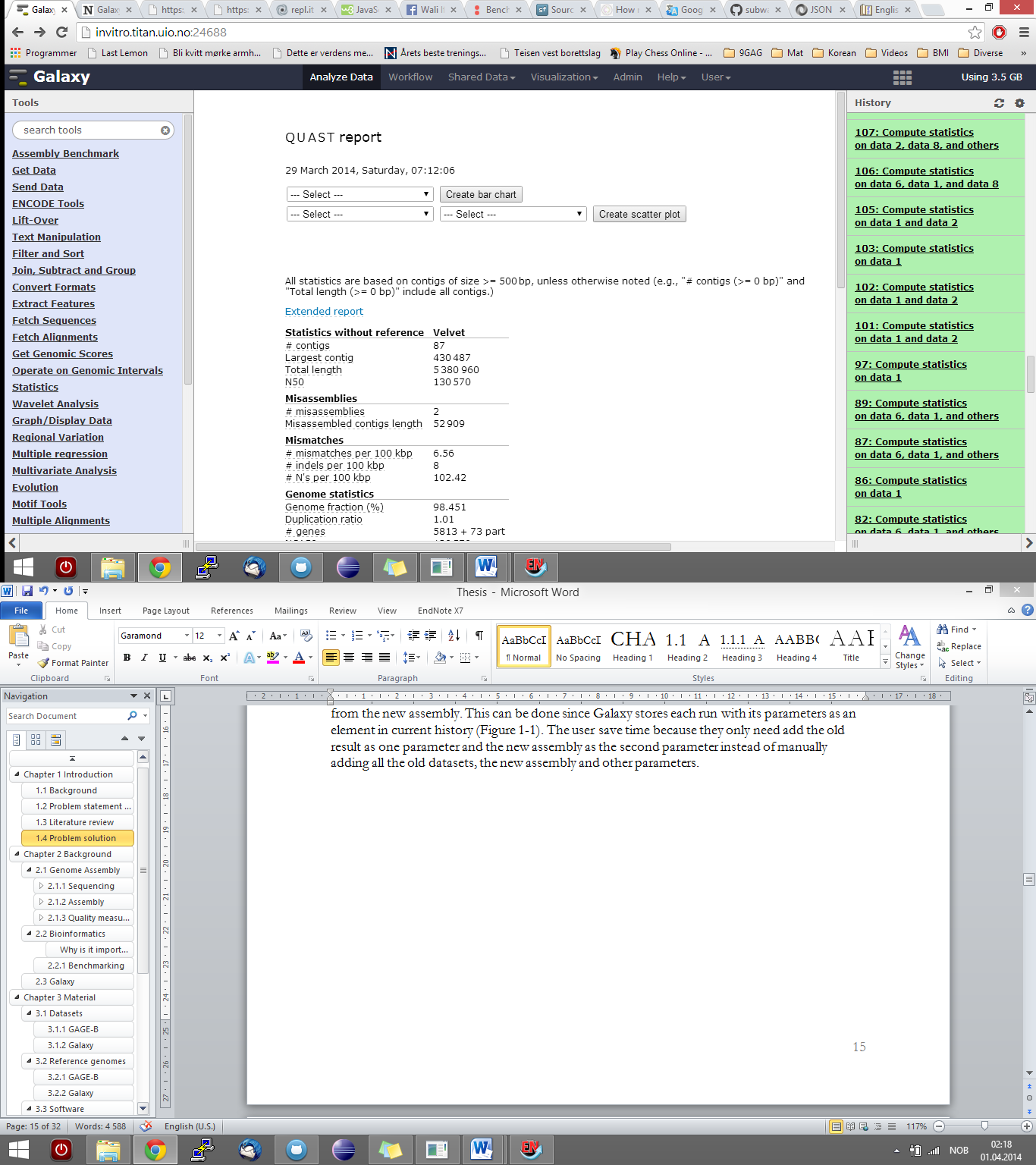


Figure 1‑ Screenshot of the Galaxy tool with an example of the history panel

# Background

Since the thesis will involve discussion of problem areas in bioinformatics that require some biological knowledge, this chapter will provide the basics of genome assembly, benchmarking in bioinformatics and the Galaxy Project.

## Genome Assembly

Genome assembly is the process of trying to reconstruct a sequence from smaller sequences called reads. This process could be trivial if each read had a unique placement, but it doesn’t, thus making assembly quite a difficult process.

### Sequencing

When looking at sequencing in a biological context, it is usually referred to as a process (a method or technology) that is used to determine the order of nucleotides in a DNA molecule. This process can be used to determine the order of genes, full chromosomes or entire genomes. This is important because the sequence in which the nucleotides appears in gives scientists valuable information about that part of the DNA which can, for instance, be used to look for disease-causing mutations in genes.

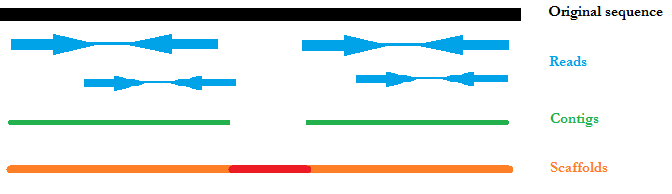


Figure 2‑ – Simplified sequencing structure

Sequencing can, roughly speaking, be described as a process where some target DNA is shredded into reads that, with a minimum of X coverage, are used to make contigs, which are then used to make scaffolds. The goal is to end up with one continuous sequence equal to the target DNA. The new sequence can then be mapped back to a reference, if one exists, to check the correctness of the assembly to a certain degree.

The two major sequencing technologies, being de novo sequencing and next-generation sequencing, are briefly described in the next two sections.

#### De novo sequencing

De novo sequencing (from latin as “from the beginning”, “afresh” or “anew”) is a collective term used for methods that sequence a DNA sequence for the first time (meaning that it has no previously known sequence), known genomes where significant structural variation is expected, or for microbial sequencing that includes experimental strains and genomes with high plasticity. This kind of sequencing use starting materials such as bacterial, viral, phage, fungus, eukaryote genomic DNA, fragmented DNA or BACs and fosmids. It can use human genome, but substantial challenges exist for its application to this.

**Sanger and shotgun sequencing**

Shotgun sequencing was introduced by Sanger and other colleagues in 1977 and was the first mainstream sequencing technology that remained the leading genome sequencing method for nearly 25 years. The name came from the analogy of a rapidly expanding, random firing pattern of a shotgun. It obtains random sequence reads from a genome and assembles them into contigs on the basis of sequence overlap. It is straightforward for simple genomes with few repeat sequences and more challenging for complex genomes because of false overlap in the repeated regions. Even though it is widely used for smaller-scale projects and for obtaining especially long contiguous DNA sequence reads (>500 nucleotides) it has been supplanted by next generation sequencing methods for large-scale, automated genome analysis. [3]

#### Next generation sequencing

Next-Generation sequencing (NGS) technologies are usually applied as a general term for sequencing platforms that use post-Sanger technology to sequence DNA fragments in parallel. They were developed to reduce weaknesses in de novo sequencing. As a result, this technology had a major increase in efficiency as well as a decrease in cost per base and could produce a lot of data cheaply compared to de novo sequencing. The “down side” was that the read length was shorter, thus creating an assembly challenge [ ] as it made for example sequencing repeated elements difficult and could result in mis-assembled genomes as Figure X [ ] shows below.

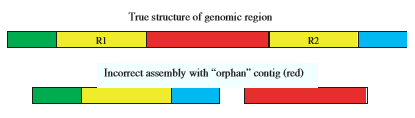


Figure 2‑ – The challenge when sequencing repeated elements  
SOURCE: STEVEN L. SALZBERG AND JAMES A. YORKE   
BEWARE OF MIS-ASSEMBLED GENOMES FIG.1

### Assembly

#### Assembly algorithms

#### Assembly challenges

As we have seen in the section above about assembly algorithms, quite a number of papers describe and discuss upon assembly methods, algorithms and how the assembly process works in general. But unfortunately, when it comes to the number of papers describing how (or discussing why) assembly is a difficult process, and challenges regarding it, the number decreases a lot. This contributes to making the understanding of genome assembly even more difficult for someone entering the field for the first time. Nonetheless, one paper that tries to discuss this is *Genome assembly reborn: recent computational challenges* by Mihai Pop,[[8](#_ENREF_8)] where Pop use solving a jigsaw puzzle as a metaphor to an assembly process. Another complementary paper used for this section is *Genome assembly forensics: finding the elusive mis-assembly* by Phillippy, Scatz and Pop[[9](#_ENREF_9)].

One of the problems regarding assembly process is about genome repeats which can be described as large stretches of sky in a jigsaw puzzle. Repeats tend to confuse the assembly process, because they seem identical to the assembler. They also make it difficult to distinguish between sequencing error and polymorphism among near-identical repeats. There are also the tandem repeats making it even more difficult. Many of the problems concerning repeats can be generalized in the following two categories:

**1. Repeat collapse and expansion**

The assembly can incorrectly gauge the number of repeats, thus including too few or too many copies, which can result in phenotypic differences, such as Huntigton’s disease.

The assembler can also mis-join reads originating from distinct repeat copies into one unit or include extra copies of repeat, both which can be noted as a higher or lower density of reads.

**2. Sequence rearrangement and inversion**

The assembler can shuffle the order of multiple repeat copies, which could be misinterpreted as a biological rearrangement event, meaning that one could draw wrong conclusions depending on the rearranged sequence.

During both repeat collapse and rearrangement, reads may get placed in a wrong copy of a repeat, therefore SNP could be a useful indicator of such a mis-assembly.

Another problem is looking at the complexity of an assembly, which depends on the number of reads being assembled. An assembly becomes more complex the more reads it assembles, making it quite complex, considering the development of shorter reads and sequencing tools that generates several million reads in a reasonable time and at a low cost.

One of the most time consuming task is probably the computation of overlaps. This task can have assembly errors which can occur due to limitations of the assembly algorithm, or by providing incorrect or incomplete assembly-parameters. It can be difficult to see where there are indels (an insertion or deletion of bases), mis-join, or find the exact placement of reads, and the detection of these errors are what scientists try to improve.

### Quality measures

There are many traps to avoid when it comes to assembly, and how well they are avoided can be measured and used to determine how well the results are. Some quality measures are easier to assess than others, especially with a reference genome. Of course, with a reference, the solution is already there, and the interesting part might be to spot the differences, compared to "normally" when the correctness of an assembly is undefined. It is therefore many criterias that can be used to assess the quality and correctness of an assembly such as the coverage and length of contigs or scaffolds, the length of the gaps between scaffolds, Nx (usuallyN50), how accurate or correct the sequence is compared to its reference, the error rate or how fast and cost-efficient it is, to mention some of the criterias. Other metrics such as the number of unaligned contigs, relocations, translocations and inversions can also be used by comparing to a reference genome. It is also possible to measure by metrics such as; the total number of contigs in the assembly, how long the assembly is (in number of bases), how long the misassembled contigs are or by looking at the (average) number of indels after x number of aligned bases.

Different measures can weight different depending on what the purpose of the assembly is. For instance, the size of scaffolds might be less important than the error rate in one case whereas the size of scaffolds might be crucial in another case. The problem is just that there’s not much that assess an assembly based on some kind of weighting. There is a statistical tool that measures some qualities, but it is rather mathematical giving only the countable/calculable scores without a proper weigth-adjustment. The next section will give a brief overview of this tool.

#### Quast

Quality Assessment Tool (QUAST) is a tool that evaluates and compares genome assemblies both with and without a reference genome. It is designed to improve existing assembly comparison software (such as GAGE) and produces results as reports, summary tables and plots that support SVG, PNG and PDF formats. An example of a metric that QUAST use is the NGx, which is like the Nx, but instead of comparing to the assembly length, the contigs are compared to the reference genome length. [15] As you can see in the figure below, quast gives a rather numerical report without giving the overall “best assembly” in the comparison.

Screenshot of quast-result html

## Bioinformatics

In N.M. Luscombe, D. Greenbaum and M. Gerstein’s review paper What is bioinformatics? An introduction and overview of the field (2001) [13] it is stated that bioinformatics is:

*“… conceptualising biology in terms of molecules (in the sense of physical chemistry) and applying* ***"informatics techniques"*** *(derived from disciplines such as applied maths, computer science and statistics) to* ***understand*** *and* ***organise*** *the* ***information*** *associated with these molecules, on a large scale. In short, bioinformatics is a management information system for molecular biology and has many* ***practical applications****.”* (page 83)

This definition, which was submitted to the Oxford English Dictionary, is a detailed way of saying that the science of bioinformatics is the use of (computer) technology in the field of biological macromolecules. But one thing this definition lacks is the fact that it does not clarify the difference between bioinformatics and computational biology that is commonly used interchangeably. Essential Bioinformatics by Jin Xiong (2006) [14] explains the difference as:

“Bioinformatics is limited to sequence, structural, and functional analysis of genes and genomes and their corresponding products and is often considered computational molecular biology. However, computational biology encompasses all biological areas that involve computation. For example, mathematical modeling of ecosystems, population dynamics, application of the game theory in behavioral studies, and phylogenetic construction using fossil records all employ computational tools, but do not necessarily involve biological macromolecules”. (p. 4)

In other words, bioinformatics is the practical approach concerning biological data, while computational biology is the theoretical approach of computer technology in biology.

#### Why is it important?

Bioinformatics is becoming a very important field because the amount of the data to work with is becoming bigger and bigger. And to store, work and analyze all this data to get the relevant information is a computer technical challenge. Some people might agree with the writer of Essential Bioinformatics, Xiong, when he writes that the goal for bioinformatics is to better understand a living cell at a molecular level to get a “global” perspective of the cell. Even though bioinformatic is having a major impact on many areas as in, for example, knowledge-based drug design, forensic DNA analysis and agricultural biotechnology, it is also worth mentioning that bioinformaticsc has its flaws as well. And to completely rely on something that can have limited accuracy or poor-quality can result in costly mistakes if not complete failures. But in the end, regardless of its limitation, bioinformatics is a field with a great potential for revolutionizing biological research in the coming decades. (Xiong, 2006)

### Benchmarking

Benchmarking can mean a specific standard that can be measured or judged by [12]. In computing, a benchmark is running a (set of) program or other operations in order to assess the relative performance. And in our case benchmarks provide a method of comparing two or more programs, assemblers, or tools to each other to determine which one is better fit for a given case/situation.

## Galaxy

Introduction to galaxy

# Material

The materials used during the research are divided into three main sections (3.1-3.3) with two subsections each (for example 3.1.1-3.1.2). The subsections are based on the two slightly different paths towards the final product. In other words, the subsections are chosen based on whether the materials were used in (partially) reproducing the GAGE-B paper, or for testing of the new Galaxy tool.

## Datasets

### GAGE-B

The reproducing of the GAGE-B papers results was initially started with all datasets used in the GAGE-B paper. Some datasets required raw data and some required trimmed. The final list of datasets are shown in table 3-1 and table 3-2, and can be downloaded from <http://ccb.jhu.edu/gage_b/datasets/index.html>. The datasets were first assembled according to the recipe on their site[[1]](#footnote-1) with an assembler named Velvet. Out of the 8 assemblers used by the GAGE-B researchers, Velvet was chosen first because it was the most easy-to-install assembler available in regards to dependencies and student permission on the faculty-computers.

|  |  |  |
| --- | --- | --- |
|  | Sequencing technology | Size |
| Aeromonas hydrophila SSU | Hiseq | 7 GB |
| Bacillus cereus VD 118 | Hiseq | 7 GB |
| Bacillus cereus ATCC 10987 | Miseq | 2 GB |
| Bacteroides fragilis HMW 615 | Hiseq | 7 GB |
| Mycobacterium abscessus 6G-0125-R | Hiseq | 2.5 GB |
| Mycobacterium abscessus 6G-0125-R | Miseq | 2 GB |
| Rhodobacter sphaeroides 2.4.1 | Hiseq | 4.5 GB |
| Rhodobacter sphaeroides 2.4.1 | Miseq | 1.5 GB |
| Staphylococcus aureus M0927 | Hiseq | 4.5 GB |
| Vibrio cholerae CO 1032(5) | Hiseq | 2 GB |
| Vibrio cholerae CO 1032(5) | Miseq | 1.5 GB |
| Xanthomonas axonopodis pv. Manihotis UA 323 | Hiseq | 8 GB |

Table ‑ – Species, their sequencing technology and size  
Downloadlink: <http://ccb.jhu.edu/gage_b/datasets/index.html>

After a while, realizing the amount of time and effort needed to reproduce the results fully, the focus shifted from all datasets and assemblers, to one set of MiSeq and HiSeq data assembled using all the 8 assemblers instead. The dataset that was chosen for this was the specie *Vibrio cholera.* The reason behind this choice was that Vibrio cholerae consisted of the smallest set of MiSeq and Hiseq data, with a total of 3.5 GB compared to a total of 4.5-8 GB for the other species. The choice was based on the fact that assembling the data with multiple assemblers, interpreting the results and comparing them to the original GAGE-B results, as described in the paper, were all quite time-consuming tasks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ABySS | CABOG | MIRA | MaSuRCA | SGA | SOAPdenovo | SPAdes | Velvet |
| V.cholearea – Hiseq | clean | clean | raw | raw | clean | clean | clean | clean |
| V.cholearea – Miseq | clean | raw | clean | clean | clean | raw | clean | clean |

Table 3‑ – Read type used for each assembler on V.cholerae

Originally, the thought behind doing the assemblies when precompiled results were available was that the results were going to be used later on in the Galaxy tool. The tool could assess the results and see if they correlated with the GAGE-B conclusion. But unfortunately, it did not go as planned and the task of partially reproducing the GAGE-B paper had to be discarded. Chapter 4 or 5 elaborate more on the reason behind this undesired choice of termination.

### Galaxy

The testing of the Galaxy tool was with the use of all assemblies provided on the following GAGE-B site: <http://ccb.jhu.edu/gage_b/genomeAssemblies/index.html>

This includes assemblies from all the species shown in Table 3-1, with the 8 assemblers mentioned in the GAGE-B paper, and with both contigs and scaffolds files.

## Reference genomes

### GAGE-B

While partially reproducing the GAGE-B results, the fastest way was to skip the reference genomes initially to check if the basic statistic was somewhat similar. The idea was to use reference genomes afterwards, but this task never advanced enough to include all the reference files so in the end, only the reference for Vibrio cholera were used. Table 3-2 below shows the details around the reference genome.

|  |  |  |
| --- | --- | --- |
| Vibrio cholerae O1 biovar eltor str. N16961 | | |
| Chromosome 1 | 2.9 MB | NC\_002505 |
| Chromosome 2 | 1.0 MB | NC\_002506 |

Table 3‑ – Reference genome used for V.cholerae  
Dowloadlink: [ftp://ftp.ncbi.nih.gov/genomes/Bacteria/  
Vibrio\_cholerae\_O1\_biovar\_El\_Tor\_N16961\_uid57623](ftp://ftp.ncbi.nih.gov/genomes/Bacteria/Vibrio_cholerae_O1_biovar_El_Tor_N16961_uid57623)

### Galaxy

The reference genome and the genefiles used for the assessment of the Galaxy tool where all downloaded from the GAGE-B’s website <http://ccb.jhu.edu/gage_b/datasets/index.html>. Each specie had quite a list of files available, but for only the sequence files (fna) and their corresponding gene files (gff) were used. The name of the reference genomes and link to their download site are shown in table 3-4.

|  |  |  |
| --- | --- | --- |
| Aeromonas hydrophila ATCC 7966 | | |
| Chromosome 1 | 4.6 MB | NC\_008570 |

|  |  |  |
| --- | --- | --- |
| Bacillus cereus ATCC 10987 | | |
| Chromosome 1  Plasmid pBc10987 | 5.1 MB  206 kB | NC\_003909  NC\_005707 |

|  |  |  |
| --- | --- | --- |
| Bacteroides fragilis 638R | | |
| Chromosome 1 | 5.2 MB | NC\_016776 |

|  |  |  |
| --- | --- | --- |
| Mycobacterium abscessus | | |
| Chromosome 1  Plasmid 1 | 4.9 MB 23.2 kB | NC\_010397 NC\_010394 |

|  |  |  |
| --- | --- | --- |
| Rhodobacter sphaeroides 2.4.1 | | |
| Chromosome 1 Chromosome 2  Plasmid A Plasmid B Plasmid C Plasmid D Plasmid E | 3.1 MB 934 kB 113 kB 113 kB 104 kB 100 kB 36.8 kB | NC\_007493 NC\_007494 NC\_009007 NC\_007488 NC\_007489 NC\_007490 NC\_009008 |

|  |  |  |
| --- | --- | --- |
| Staphylococcus aureus SA300\_TCH1516 | | |
| Chromosome 1  Plasmid pUSA300HOUMR Plasmid pUSA01-HOU | 2.8 MB 26.9 kB 3.2 kB | NC\_010079 NC\_010063 NC\_012417 |

|  |  |  |
| --- | --- | --- |
| Vibrio cholerae O1 biovar eltor str. 16961 | | |
| Chromosome 1  Chromosome 2 | 2.9 MB 1.0 MB | NC\_002505 NC\_002506 |

|  |  |  |
| --- | --- | --- |
| Xanthomonas axonopodis pv. Citrumelo | | |
| Chromosome 1 | 4.8 MB | NC\_016010 |

Table 3‑ Reference genome for each dataset  
Source: <http://ccb.jhu.edu/gage_b/datasets/index.html>

## Software

### GAGE-B

The various pieces of software used while reproducing the GAGE-B papers results were:

* ABySS v1.3.4
  + <http://www.bcgsc.ca/platform/bioinfo/software/abyss>
* CABOG v7.0
  + <http://sourceforge.net/apps/mediawiki/wgs-assembler>
* MIRA v3.4.0
  + <http://www.chevreux.org/project_mira.html>
* MSRCA v1.8.3
  + <http://www.genome.umd.edu/>
* SGA v0.9.34
  + <https://github.com/jts/sga/wiki>
* SOAPdenovo2 v2.04 + GapCloser v1.12
  + <http://sourceforge.net/projects/soapdenovo2>
  + <http://soap.genomics.org.cn/soapdenovo.html>
* SPAdes v2.3.0
  + <http://bioinf.spbau.ru/en/spades>
* Velvet v1.2.08
  + <http://www.ebi.ac.uk/~zerbino/velvet/>

### Galaxy

Software used in combination with the Galaxy framework was:

* Python v2.7
* Quast v2.2
* Google charts

# Methods

This chapter covers a description of the tool and implementation.

## Reproducing the GAGE-B results

After a while, realizing the amount of time and effort needed to reproduce the results fully, I decided that I would focus on just one species.

Why vibrio cholera? Because of this and that.

## Galaxy

# Results

Describe why you could not use your own assembly based on the GAGE-B datasets.

# Discussion

## Interpreting the GAGE-B results

## Analysis of galaxy tool

### Performance

### Potential use

### Weakness

## Further work

## Conclusion

Referanseliste!

Glossary =)

Glossary

|  |  |
| --- | --- |
|  |  |
|  |  |
| BAC | Abbreviation of *bacterial artificial chromosome* |
| bacterial artificial chromosome (BAC) | An artificially constructed segment of nucleic acid used for transforming and cloning in bacteria, usually E. coli. |
| bacteriophage | A virus that parasitizes a bacterial cell |
| chromosome | A threadlike, gene-carrying structure found in the nucleus. Each chromosome consists of one very long DNA molecule and associated proteins |
| contig | A continuous sequence of DNA that have been assembled from overlapping reads |
| coverage | The average number of reads representing a given nucleotide in a reconstructed sequence; also known as read depth or depth |
| de novo sequencing |  |
| deoxyribonucleic acid (DNA) | A double-stranded, helical nucleic acid molecule capable of replicating and determining the inherited structure of a cell's proteins |
| DNA | Abbreviation of *deoxyribonucleic acid* |
| eukaryote | An organism whose cells contain membraine-bound organelles and whose DNA is enclosed in a cell nucleus and is associated with proteins |
| fungus | About 80,000 known species of organisms of the kingdom Fungi, which includes the yeasts, rusts, smuts, mildews, molds, mushrooms, and toadstools |
| genome | The complete complement of an organism's genes; an organism's genetic material |
| inversion | A misjoin of a scaffold/contig where the two pieces map to the opposite strands on the same chromosome |
| nucleus | The chromosome-containing organelle of a eukaryotic cell |
| phage | A virus that infects bacteria; also called a bacteriophage |
| polymorphism |  |
| protein | A three-dimensional biological polymer constructed from a set of 20 different monomers called amino acids |
| read | Pieces of a sequence acquired under sequencing used for mapping/assembly that vary in length from less than 100 base pairs up to several thousand base pairs. Usually, with a double stranded chain, the reads contains the direction as well |
| read depth | See *coverage* |
| relocation | A misjoin of a scaffold/contig where the two pieces map to different locations on the reference genome |
| scaffold | A series of contigs that are in the right order but not necessarily connected in one continuous stretch of sequence. The remaining gaps between contigs in a scaffold can usually be sequenced because the placement of contigs are often known |
| translocation | A misjoin of a scaffold/contig where the two pieces map to different chromosomes or plasmids |
| viral | A biological virus |
| virus | A submicroscopic, noncellular particle composed of a nucleic acid core and a protein coat (capsid); parasitic; reproduces only within a host cell. |
|  |  |

Appendix a  
(Plot/Figure)



figure ‑

Appendix B  
(Tables)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table ‑ Bla bla bla

1. Miller, J.R., A.L. Delcher, S. Koren, et al.,

*Aggressive assembly of pyrosequencing reads with mates.*

Bioinformatics, 2008. **24**(24): p. 2818-2824.

2. Zerbino, D.R. and E. Birney,

*Velvet: Algorithms for de novo short read assembly using de Bruijn graphs.*

Genome Research, 2008. **18**(5): p. 821-829.

3. Simpson, J.T., K. Wong, S.D. Jackman, et al.,

*ABySS: a parallel assembler for short read sequence data.*

Genome Research, 2009. **19**(6): p. 1117-23.

4. Gurevich, A., V. Saveliev, N. Vyahhi, et al.,

*QUAST: quality assessment tool for genome assemblies.*

Bioinformatics, 2013. **29**(8): p. 1072-5.

5. Salzberg, S.L., A.M. Phillippy, A. Zimin, et al.,

*GAGE: A critical evaluation of genome assemblies and assembly algorithms.*

Genome Research, 2012. **22**(3): p. 557-67.

6. Bradnam, K., J. Fass, A. Alexandrov, et al.,

*Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species.*

Gigascience, 2013. **2**(10).

7. Earl, D., K. Bradnam, J. St John, et al.,

*Assemblathon 1: a competitive assessment of de novo short read assembly methods.*

Genome Research, 2011. **21**(12): p. 2224-41.

8. Pop, M.,

*Genome assembly reborn: recent computational challenges.*

Briefings in bioinformatics, 2009. **10**(4): p. 354-66.

9. Phillippy, A.M., M.C. Schatz, and M. Pop,

*Genome assembly forensics: finding the elusive mis-assembly.*

Genome Biology, 2008. **9**(3).

1. <http://ccb.jhu.edu/gage_b/recipes/recipes.pdf> [↑](#footnote-ref-1)