UIO forside her

aspects of reproducability and reusability of genome assembly evaluation with the gage-b as a case study

Sabba Ifzal

July 1, 2014

Acknowledgements

Takke folk som har hjulpet til

Veiledere – Torbjørn, Lex, Geir Kjetil og Ksenia

Andre folk i 10 – Sveinung, Kai, Jon, Morten

Ruth for sensur

Ghada for samtaler

etc

Abstract

Kort oppsummert hele oppgaven

Why you do this

How you do this

The results

Conclusion

Preface

This master’s thesis was written with a masterstudent as intended reader. The reader is expected to have basic knowledge about biology, such as DNA/RNA and the principles surrounding genomes and inheritance.

Contents

[Chapter 1 Introduction 1](#_Toc388041429)

[1.1 Background 1](#_Toc388041430)

[1.2 Problem statement / Aims 1](#_Toc388041431)

[1.3 Literature review 2](#_Toc388041432)

[1.4 Problem solution 2](#_Toc388041433)

[Chapter 2 Background 4](#_Toc388041434)

[2.1 Genome sequencing and assembly 4](#_Toc388041435)

[2.1.1 Sequencing 4](#_Toc388041436)

[2.1.2 Assembly 7](#_Toc388041437)

[2.1.3 Quality measures 13](#_Toc388041438)

[2.2 Reproducability 16](#_Toc388041439)

[2.3 Reusability 16](#_Toc388041440)

[2.4 The Galaxy project 17](#_Toc388041441)

[2.4.1 Galaxy mailing lists 17](#_Toc388041442)

[2.4.2 Galaxy objects 18](#_Toc388041443)

[Chapter 3 Material 21](#_Toc388041444)

[3.1 Datasets 21](#_Toc388041445)

[3.1.1 Read data 21](#_Toc388041446)

[3.1.2 Assemblies 22](#_Toc388041447)

[3.1.3 Reference genomes 22](#_Toc388041448)

[3.2 Software 23](#_Toc388041449)

[Chapter 4 Methods 28](#_Toc388041450)

[4.1 Reproducing the GAGE-B results 28](#_Toc388041451)

[4.2 Galaxy 28](#_Toc388041452)

[4.2.1 Published Galaxy-page 28](#_Toc388041453)

[4.2.2 Implementation and testing of Galaxy tools 30](#_Toc388041454)

[4.2.3 Simple user manual for the tools 33](#_Toc388041455)

[Chapter 5 Results 45](#_Toc388041456)

[Chapter 6 Discussion 50](#_Toc388041457)

[6.1 Interpreting the GAGE-B results 50](#_Toc388041458)

[6.2 Analysis of galaxy tool 50](#_Toc388041459)

[6.2.1 Performance 50](#_Toc388041460)

[6.2.2 Potential use 50](#_Toc388041461)

[6.2.3 Weakness 50](#_Toc388041462)

[6.3 Further work 51](#_Toc388041463)

[6.4 Conclusion 51](#_Toc388041464)

List of figures

[Figure 1‑1 Screenshot of the Galaxy tool with an example of the history panel 3](#_Toc390732520)

[Figure 2‑1 Basic steps of genome sequencing and assembly [8] 4](#_Toc390732523)

[Figure 2‑2 Template preparation in 454 pyrosequencing technology [10](p.33) 5](#_Toc390732524)

[Figure 2‑3 Template preparation in illumina [10](p.33) 6](#_Toc390732525)

[Figure 2‑4 OLC step 1[11] 8](#_Toc390732527)

[Figure 2‑5 OLC step 2[11] 8](#_Toc390732528)

[Figure 2‑6 Sanger sequencing (A) vs next generation sequencing (B) [14] 11](#_Toc390732529)

[Figure 2‑7 The challenge of sequencing repeated element [16](Fig 1) 12](#_Toc390732530)

[Figure 2‑8 Quast report example (http://quast.bioinf.spbau.ru/) 15](#_Toc390732532)

[Figure 2‑9 Galaxy instance home-page 18](#_Toc390732536)

[Figure 2‑10 An example of shared or published histories in galaxy 19](#_Toc390732537)

[Figure 2‑11 Example of creation of a workflow 19](#_Toc390732538)

[Figure 2‑12 Example of running the workflow from figure 2-6 20](#_Toc390732539)

[Figure 4‑1 List of published histories 28](#_Toc390732549)

[Figure 4‑2 the naming of datasets for Mycobacterium abscessus 6G-0125-R 30](#_Toc390732550)

[Figure 4‑3 (From left) tool\_conf.xml file and the tool menu in Galaxy 31](#_Toc390732551)

[Figure 4‑4 One history with test-data used in the user manual 34](#_Toc390732552)

[Figure 4‑5 Default startpage for the «Compute statistics»-tool 34](#_Toc390732553)

[Figure 4‑6 Blankpage for the tool «Upload file» 35](#_Toc390732554)

[Figure 4‑7 Screenshot of new fields available after pressing “Add new dataset” 36](#_Toc390732555)

[Figure 4‑8 Screenshot of the option “Use built in reference” 36](#_Toc390732556)

[Figure 4‑9 Screenshot of the option «Upload your reference» 37](#_Toc390732557)

[Figure 4‑10 Screenshot of the option «Use built in reference” 37](#_Toc390732558)

[Figure 4‑11 Screenshot of the option «Upload your reference» 38](#_Toc390732559)

[Figure 4‑12 Screenshot of quast specific parameters with their default value 38](file:///C:\Users\Sabba\Documents\GitHub\masterthesis\Written\Thesis\Thesis.docx#_Toc390732560)

[Figure 4‑13 Stages of tool execution 39](#_Toc390732561)

[Figure 4‑14 Example of tool output and in browser view of report.html 39](#_Toc390732562)

[Figure 4‑15 Example of interactive table in report.html 40](#_Toc390732563)

[Figure 4‑16 Predrawn plot in report.html 40](#_Toc390732564)

[Figure 4‑17 Bar chart in report.html 42](file:///C:\Users\Sabba\Documents\GitHub\masterthesis\Written\Thesis\Thesis.docx#_Toc390732566)

[Figure 4‑18 Scatterplot in report.html 43](#_Toc390732567)

[Figure 4‑19 Default startpage for the «compare statistics» tool 44](#_Toc390732568)

[Figure 4‑20 Screenshot of how to choose input data for “Compare statistic” 44](#_Toc390732569)

[Figure 4‑21 Example of tool output and in browser view of report.txt 45](#_Toc390732570)

List of tables

[Table 2‑1 Comparison of illumina sequencing instruments and capasity \* The error rate for Illumina MiSeq and HiSeq sequencer is ≥0.1% Source: http://dna.uga.edu/services/hiseq-miseq-sequencing/ 6](#_Toc388041504)

[Table 3‑1 – Species, their sequencing technology and size Downloadlink: http://ccb.jhu.edu/gage\_b/datasets/index.html 21](#_Toc388041518)

[Table 3‑2 – Read type used for each assembler on V.cholerae 22](#_Toc388041519)

[Table 3‑3 Reference genome for each dataset Source: http://ccb.jhu.edu/gage\_b/datasets/index.html 23](#_Toc388041520)

[Table 6‑1 Raw reads are produced by the sequencer 61](#_Toc388041535)

[Table 6‑2 Some Problems discovered in the GAGE-B recipe upon use 63](#_Toc388041541)

# 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, better known as an assembly, is performed by assemblers 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.

Visualize things .. edit first line.

Some biologists have pointed out that one of the disadvantages of current benchmarks is the lack of visualization, 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, define reproduce/reuse here**

## 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.

Reproduceability gage-b… and reuseability. Can we have the same approach for our (same) data

Ikke nevn Galaxy her! Målet med oppgaven er å lage et system som kan gjøre dette og dette. Ikke noe annet, ikke noe spesifikt. Eksakt hva du skal gjøre. Kobling til gage-b her, hvis det finnes en kobling.

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.

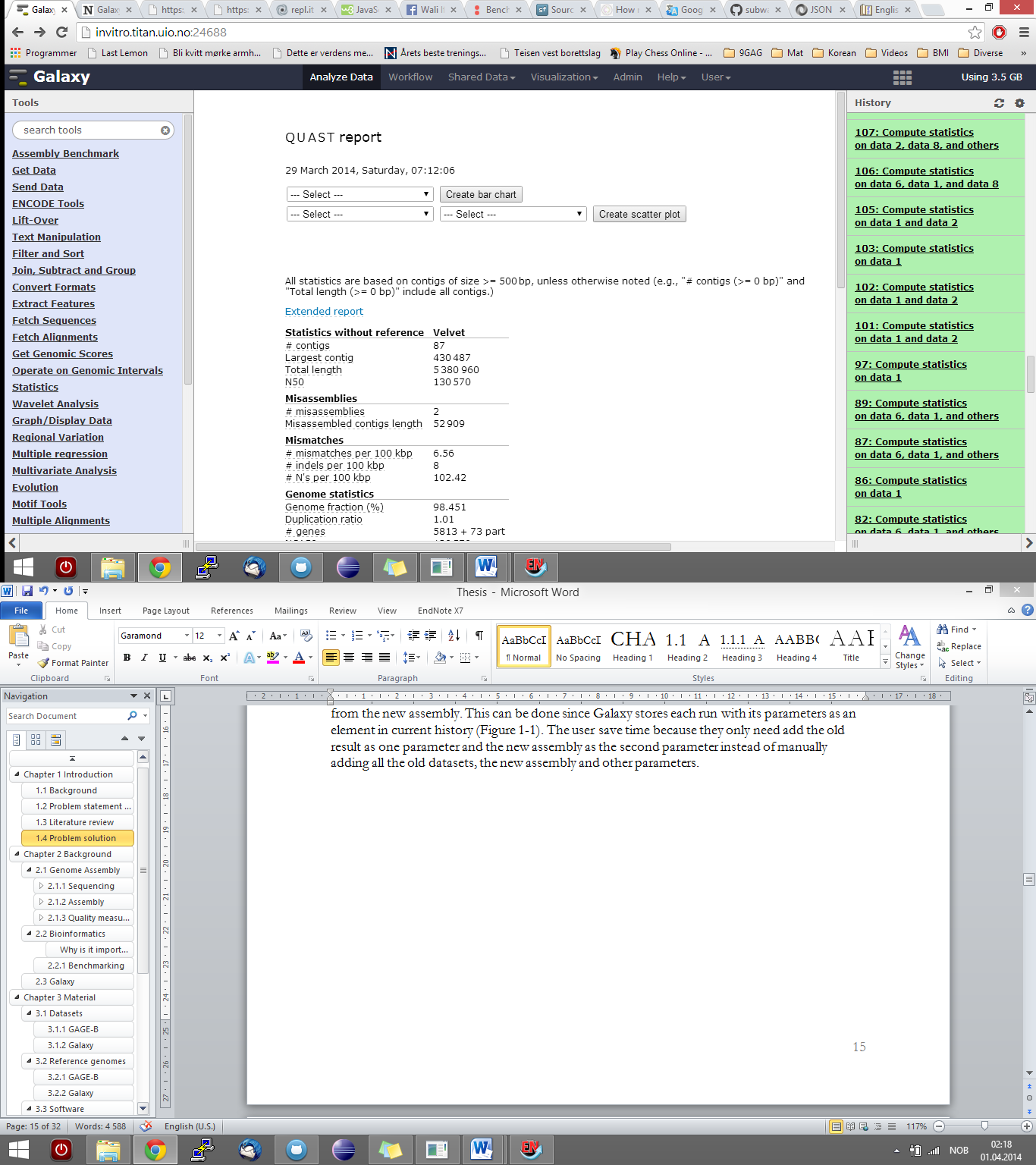


Figure 1‑1 Screenshot of the Galaxy tool with an example of the history panel

# 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, reproducibility, reusability and the Galaxy Project.

## Genome sequencing and assembly

What more powerful form of study of mankind could there be   
than to read our own instruction book?  
Francis Collins

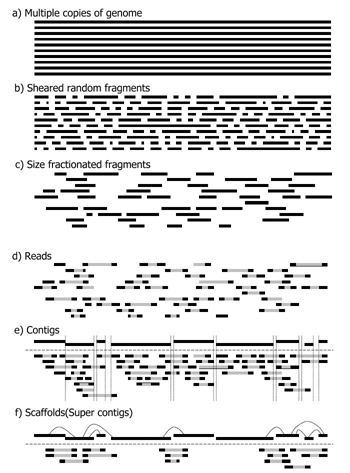


Figure 2‑1 Basic steps of genome sequencing and assembly [[8](#_ENREF_8)]

### 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 obtain a set of **reads** from one or multiple copies of a genome as illustrated in Figure 2-1(a-d). How this process works in practice depend on the sequencing technology used which will be explained in further details in the next sections.

#### Sequencing technologies

This section will cover three well-known sequencing technologies, including how they work, how many reads and bases they create. Information used in this section is gathered from the book *Algorithms in Bioinformatics* [[9](#_ENREF_9)].

**454 PYROSEQUENCING TECHNOLOGY**

Pyrosequencing is based on “sequencing by synthesis”, which means that, given a single-strand DNA template, the complementary strands can be synthesized sequentially by adding nucleotides (A, C, T and G) one by one. Light is produced with the help of chemiluminescent enzyme when the complementary nucleotide is added in the template, and this light is used to reconstruct the template sequence. The technology was licensed and developed into Next Generation Sequencing technology by 454 Life Science in 2005. With this technology, a set of single-stranded **template DNA** is loaded on a set of beads, amplified using emulsion-based **PCR** (**emPCR**) and read in parallel using pyrosequencing (as shown in Figure 2-2).

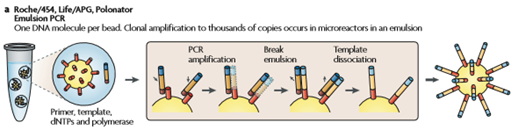


Figure 2‑2 Template preparation in 454 pyrosequencing technology [[10](#_ENREF_10)](p.33)

**ILLUMINA SOLEXA TECHNOLOGY**

This technology was acquired and commercialized by Illumina in 2006. The difference from 454 is that Illumina generated shorter reads. This technology consists of the following four steps:

1. A set of single stranded template DNA is prepared
2. The two ends of the template DNA is randomly fixed on the surface of a flow cell
3. The template DNA is amplified with **bridge PCR**
4. The template DNA is read in parallel using four-color fluorescent dye and a polymerase-mediated primer extension reaction (as shown in Figure 2-3)

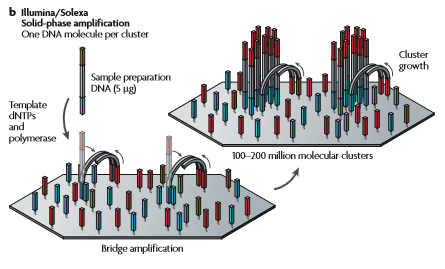


Figure 2‑3 Template preparation in illumina [[10](#_ENREF_10)](p.33)

The datasets used in this thesis are only **MiSeq** and **Hiseq** Illumina **paired-end reads**. Table 2-1 lists the differences between the instruments and also run type as paired-end reads (PE) or **single-end reads** (SE) with the read length immediately after. In general, Illumina MiSeq focus on speed and simplicity for targeted and small genome sequencing, with small genome, **amplicon**, and targeted gene panel sequencing as key applications. Illumina HiSeq on the other hand, focuses on power and efficiency for large-scale genomics, **exome**, **transcriptome** sequencing, and more.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instrument \* | Run type | Run time | Bases/read | Max number  of reads/lane (million) | Total number of bases/lane |
| MiSeq (50 cycles v2 kit) MiSeq (300 cycles v2 kit) MiSeq (500 cycles v2 kit) MiSeq (150 cycles v3 kit) MiSeq (150 cycles v2 kit) MiSeq (600 cycles v3 kit) | SE50 PE150 PE250 SE150 PE75 PE300 | 5 hours 24 hours 39 hours 24 hours 24 hours 65 hours | 50 150 + 150 250 + 250 150 75 + 75 300 + 300 | 15-17 30-40 30-40 22-25 44-50 44-50 | 750-850 MB 4.5-5.1 GB  7.5-8.5GB  3.3-3.8 GB  3.3-3.8 GB  13.2-15 GB |
| HiSeq 2000 v3 | SE50 SE100  PE50  PE100 | 2-3 days 5-6 days  5-6 days  9-11 days | 50 100  50 + 50  100 + 100 | 187 187  374  374 | 9.4 GB 18.7 GB  18.7 GB  37.4 GB |

Table 2‑1 Comparison of illumina sequencing instruments and capasity  
 \* The error rate for Illumina MiSeq and HiSeq sequencer is ≥0.1%  
 Source: <http://dna.uga.edu/services/hiseq-miseq-sequencing/>

**ABI SOLID TECHNOLOGY**

Supported Oligonucleotide Ligation and Detection Platform (SOLiD) was introduces in 2007 by Life technologies. This technology loads a set of single-stranded template DNA on beads, amplifies them using emPCR, and distributes the beads randomly on a glass plate so that it can be read in parallel via sequencing by ligation.

### Assembly

Assembly can, roughly speaking, be described as a process where some reads, with a minimum of X **read depth** or **coverage**, are used to make **contigs** (Figure 2-1(e)), which are then used to make **scaffolds** (Figure 2-1(f)). A minimum of read coverage is used to ensure the reliability of the contigs because the more reads that overlap on a given position, the safer it is presume that the given nucleotide is correct. The desired result from an assembly is to end up with one continuous sequence equal to the target DNA. The new sequence can be mapped back to a reference, if one exists, to check the correctness of the assembly. But, it is important to make sure that the differences are in fact errors and not just some kind of structural variation or mutation to avoid wrong biological conclusions. The mapping process can also be used to determine the order of genes, full chromosomes or entire genomes. This determination 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.

#### Assembly algorithms

There are many different approaches used for an assembly with the greedy algorithm being one of the first used. This approach will try to find and merge the shortest common supersequence, meaning the two fragments with the largest overlap. This process will be repeated until only one fragment is left as a suboptimal solution. This is obviously both time and resource consuming considering the amount and complexity of datasets researchers work with today, thus making the algorithm mostly abandoned. Another approach for the greedy algorithm is to use it for aligning DNA sequences that differ only by sequencing errors, or by equivalent errors from other sources, where it can be much faster than traditional dynamic programming approaches and yet produce an alignment that is guaranteed to be theoretically optimal.

<http://online.liebertpub.com/doi/abs/10.1089/10665270050081478>

Today, the two most used approaches for assemblers are Overlap – Layout - Consensus (which is used by Celera Assembler CABOG) and de Bruijn graph (which is used by ABySS and Velvet) <http://bfg.oxfordjournals.org/content/11/1/25.full> . The Overlap – Layout – Consensus is a well-established and powerful method, but it can be a challenge to implement correctly. In general, the idea behind OLC is simple. There are three steps to this approach:

1. Overlap
   1. This step is the so-called “computation-step” meaning that this is where the overlaps are found by aligning the sequence of the reads. The overlaps are initiated by k-mer seeds like this:



Figure 2‑4 OLC step 1[[11](#_ENREF_11)]

1. Layout
   1. This is the step with the graph simplification. Here it layout the reads based on the alignment. By now, the overlap-step has finished aligning the sequence of reads which can be presented as a graph:



Figure 2‑5 OLC step 2[[11](#_ENREF_11)]

* 1. As seen from nicely colored illustrations in both steps above, the graph simplification of step one makes it easier to understand how the reads are structured. There are two different roads from R4, which can indicate that the sequence R1-R2-R3-R4 might actually be repeats in the original sequence.

1. Consensus
   1. Step three; get consensus by joining all sequences of reads, merging overlaps that result in the final sequence.

De Bruijn graph on the other hand, is newer and easier to implement than the OLC method. The drawback is that it can be a bit problematic for complex genomes since it is based on short words (k-mers), but it is ideal for high coverage, short read data. <http://genome.cshlp.org/content/18/5/821.short> This graph theory was actually developed outside the field of bioinformatics as a mathematical concept developed for use with a small alphabet of a limited size. It has later on been adapted in the field of biology which operates with nucleotides as a small alphabet with the four letters A, T, C and G. It is considered by many researchers to be the best solution for short reads smaller than 100 nucleotides. The idea is kind of similar to the OLC approach, but there are still some differences that are important to understand.

Instead of having k-mer seed like in OLC, de Bruijn graph uses small k-mers and iterates through the reads, base by base, and obtains all the “k-words” available in the sequence. For instance, if we have the following read: GGACCTACA and TACAAAT and uses k-mers of length 3, then the k-word (in bold) will be computed like this:

**READ 1 READ 2**

**GGA**CCTACA **TAC**AAAT  
G**GAC**CTACA T**ACA**AAT  
GG**ACC**TACA TA**CAA**AT  
GGA**CCT**ACA TAC**AAA**T  
GGAC**CTA**CA TACA**AAT**GGACC**TAC**A  
GGACCT**ACA**

The k-words are then matched across reads to find overlap and the matches are used to create a k-word graph containing multiple nodes with unique k-words. In this example, the result could be something like this:

**GGA -> GAC -> ACC -> CCT -> CTA -> TAC -> ACA -> CAA -> AAA -> AAT  
 G G A C C T A C A AAT**

Where the red nodes represent k-words from read1, the blue nodes represent k-words from read 2, and the purple nodes represent k-words where the two reads overlap, resulting in the sequence in green.

#### 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,[[12](#_ENREF_12)] where he 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[[13](#_ENREF_13)].

One of the problems regarding assembly process is genome repeats which can be described as large stretches of blue 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. Assemblers also have to deal with the difficulties of having a sequence with tandem repeats. 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 generate 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.

#### Sanger sequencing vs next generation sequencing

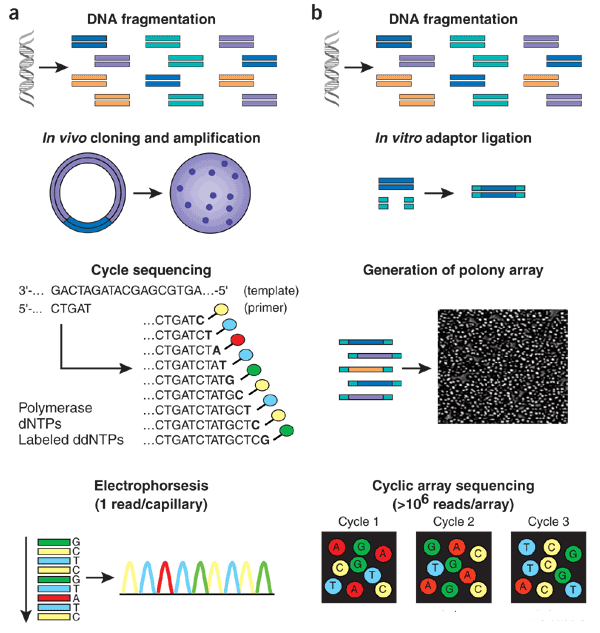


Figure 2‑6 Sanger sequencing (A) vs next generation sequencing (B) [[14](#_ENREF_14)]

The first method of sequencing the genetic code, known as Sanger sequencing or chain termination sequencing, was developed by Fred Sanger in the late 70’s. The general outline of this sequencing method starts with separating the DNA sequence into two strains followed by **in vivo** cloning and amplification using chemically altered bases. A sequencing primer is attached to the single stranded DNA to get a starting point and the chemically altered bases causes the amplification process to stop each time one particular letter is incorporated into the growing DNA chain. This process is carried out for all four bases, and then the fragments are put together by examining the relative positions of the chain termination products to reveal the sequence of the original piece of DNA.

Next-Generation sequencing (NGS) technologies are usually applied as a general term for sequencing platforms that use post-Sanger technology to sequence DNA fragments. The demand for low-cost sequencing resulted in the development of NGS technologies that parallelize the sequencing process, producing thousands or millions of sequences concurrently. As a result, this technology had a major increase in efficiency as well as a decrease in cost per base. The drawback was that the read length was shorter, thus creating an assembly challenge [[15](#_ENREF_15)] as it made, for instance sequencing repeated elements difficult and could result in mis-assembled genomes as Figure 2-4 shows. A more detailed explanation about the difficulties with repeated elements can be read in chapter 2.1.2s subsection *Assembly challenges*.

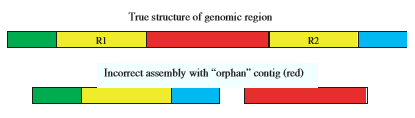


Figure 2‑7 The challenge of sequencing repeated element [[16](#_ENREF_16)](Fig 1)

In NGS technologies, what happens?

The differences between Sanger sequencing vs Next generation sequencing in terms of sampling, tracking and data relationship is further described in Table 2-2.

|  |  |  |
| --- | --- | --- |
|  | Sanger | Next Generation |
| Sequencing samples | Clones, PCR | DNA libraries |
| Sample tracking | Many samples in 96, 384 well plates | Few |
| Preparation steps | Few, sequencing reactions clean up | Many, complex procedures |
| Data collection | Samples in plates 96, 384 | Samples on slides 1- 16+ |
| Data | One read / sample | Thousands and millions reads /sample |

Table 2‑2 The differences between sanger and next generation sequencing [[17](#_ENREF_17)]

#### De novo sequencing vs resequencing

De novo sequencing (from latin as “from the beginning”, “afresh” or “anew”) is a collective term used for:

* Methods that sequence unknown genomes or when no reference sequence is available
* Methods that sequence known genomes where significant structural variation is expected
* Microbial sequencing that includes experimental strains and genomes with high plasticity

This kind of sequencing use a variety of starting materials including:

* Bacterial
* Viral
* Phage
* Fungus
* BACs
* Fosmids
* Eukaryote genomic DNA
* Fragmented DNA

Resequencing on the other hand can be used to catalogue sequence variation and understanding the biological consequences of them. It is a key step in detection of mutations associated with various congenital diseases and the techniques can be divided into those which test for known mutations (genotyping) and those who look for mutations in a given target region (variation analysis).

The typical mutations being tested are:

* *Substitutions*, also known as single nucleotide polymorphism (SNP), where a single nucleotide (A, T, C, G) differs between members of a biological species or paired chromosomes
* *Insertions*, which can be an incorrectly addition of one or more nucleotide base pairs into a DNA sequence or an incorrectly addition of a section from one chromosome to another
* *Deletions*, where a part of a sequence or chromosome is missing. The deleted size can be anywhere from a single base pair to an entire piece of a chromosome.

### 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 number of genes might be crucial in another case. A couple of commonly used quality measures are listed in the subsections below, followed by a brief overview of a tool that assesses an assembly using these measures among others.

#### Nx

Nx of an assembly is a metric defined as a weighted median of the lengths of the sequences it contains, equal to the length of the longest sequence s, such that the sum of the lengths of sequences greater than or equal in length to s is greater than or equal to x% of the genome being assembled [[7](#_ENREF_7)].

#### NGX

NGx of an assembly is a metric similar to Nx except that it accounts for x% of the length of the reference genome, instead of x% of the assembly length [[4](#_ENREF_4)].

#### Coverage

The read coverage can also say something about the assembly. Usually, the higher the coverage is, the more politely is the nucleotide base position. The most ideal situation is when all the contigs have as high read coverage as possible.

#### Contig size

This is defined as the total number of contigs in an assembly. In general, the fewer and longer the contigs are, the better it is. That is of course while assuming that the contigs are assembled correctly, which unfortunately is not true in all cases. This is where the coverage might clarify the correctness of the contigs.

#### Representation

The representation can be used as a quality measure, assuming that a reference genome is available. It is then defined as the percent of the reference genome which is covered by assembled contigs. This is a measure that is desired to be as high as possible.

#### Number of gaps

This measure calculates the number of gaps present between assembled contigs which align to the reference. Since misassemblies can force a contig to be aligned in more than one location, the number of gaps could be higher than the total number of contigs [[18](#_ENREF_18)].

#### Mismatch rate

The mismatch rate is the percent of mismatches in the assembly. The percentage is calculated by dividing the total number of mismatches by the total number of bases in the specified reference or reference region[[18](#_ENREF_18)].

#### Indel rate

This **indel** rate is defined as the percent of indels in the assembly. The percentage is obtained by dividing the total number of indels by the total number of bases in the specified reference or reference region [[18](#_ENREF_18)].

#### Misassembly rate

The misassembly rate is defined as the average number of misassemblies found per Mb of sequence [[18](#_ENREF_18)]. A position is considered a misassembly if it is a **relocation**, a **translocation** or an **inversion**.

#### Error rate

The error rate is defined as the percent of errors in an assembly. The rate is obtained by dividing the total number of base errors (indel and mismatch) by the total number of bases in the specified reference or reference region [[18](#_ENREF_18)].

#### 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 [[4](#_ENREF_4)] As you can see in the figure below, quast gives a rather numerical report without giving the overall “best assembly” in the comparison. QUAST is rather mathematical, thus giving the user a table with numeric data and a minimum of dynamic and static plots based on the table-values.

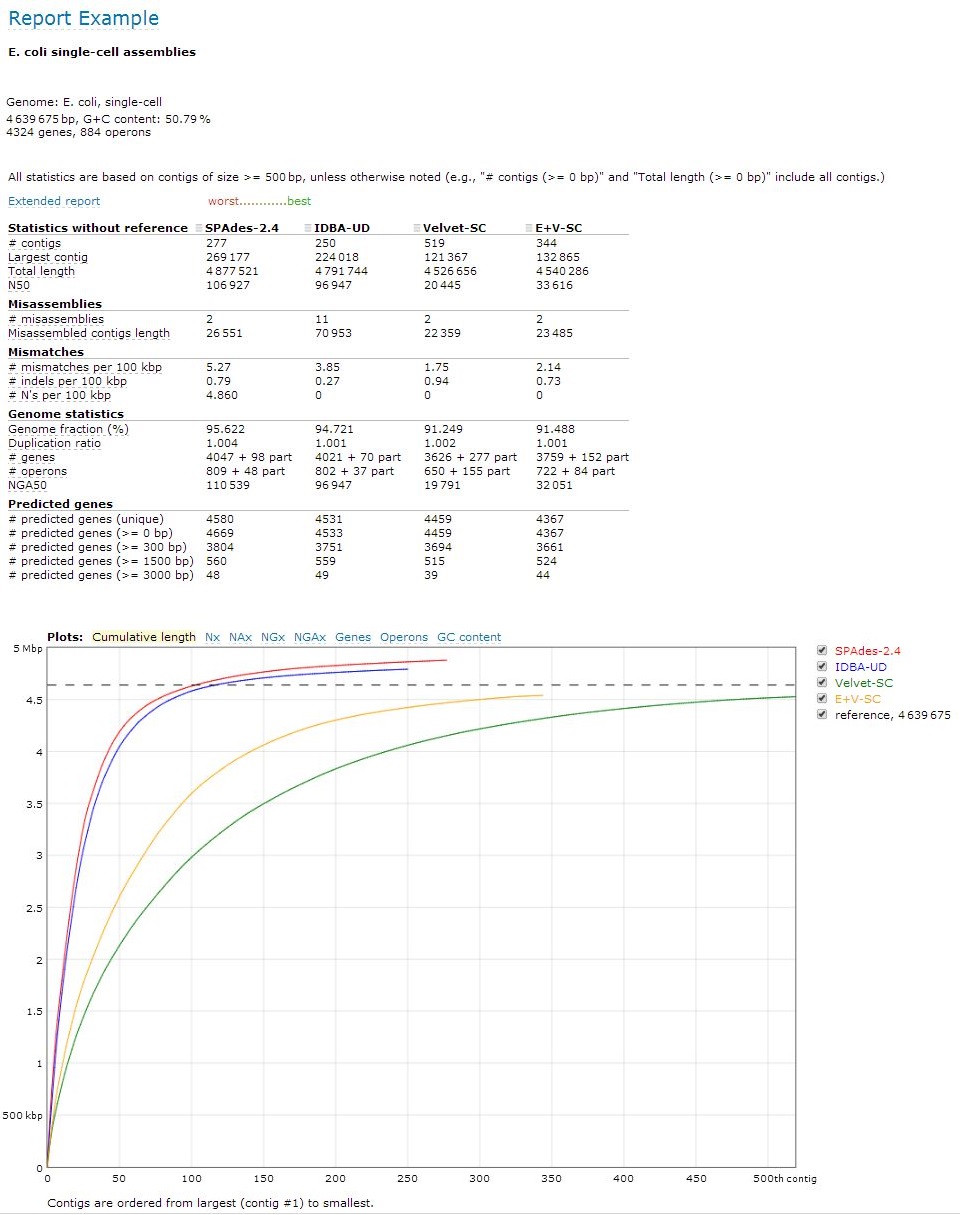


Figure 2‑8 Quast report example (<http://quast.bioinf.spbau.ru/>)

## Reproducability

An experiment is reproducible until another laboratory tries to repeat it.   
Alexander Kohn [[19](#_ENREF_19)]

One of the main principles of **the scientific method** is the ability to reproduce an entire experiment or study. Reproducibility is said to be a fundamental part of science because it enables people to develop work further by applying new data or methodology, build on the work of others or to verify published results. It is expected, in a biological context, that findings can be replicated by independent data, analytical methods, laboratories and instruments. [[20](#_ENREF_20)]

Unfortunately, in the field of bioinformatics, the amount and complexity of data collections with the increasingly sophisticated analyses can sometimes make it difficult to reproduce the results fully. In some cases, studies cannot be replicated at all due to the lack of time, money or resources while in other cases, even if there exits somewhat reproducible research, the documentation is poorly written, making a correct reproduction quite difficult. The documentations might be written poorly because the researchers feel that they need to sustain their reputation by getting results fast so that they can win the race of publishing new findings first. Unfortunately, this often implies that the end justifies the means, making reproducibility quite difficult. Lately, to avoid those kinds of trouble for other, maybe independent researchers, it has been common to provide the datasets and software used for the findings so that other scientist can verify the published findings or conduct alternative analysis.

Many papers have been written over the years about reproducibility and one paper written by Sandve et al. [[21](#_ENREF_21)] have a good 10-rules description for reproducible computational research as follows:

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analysis that includes randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing details to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

The replication of findings and studies by multiple independent scientists will in the future be important to the accumulation of scientific evidence. Hopefully, more researchers will adapt to this description in upcoming publications, thus making reproducibility simpler.

## Reusability

Good programmers know what to write. Great ones know what to rewrite (and reuse).  
Eric S. Raymond [[22](#_ENREF_22)]

The notion of reusability is, as stated by Prieto-Diaz in *Status Report: Software reusability* [[23](#_ENREF_23)], an old idea were solutions to current problems are modified, combined, and adapted to solve similar new problems. In computer science and software engineering, reusability is described as the reuse of source code segments, product generated during software development (such as system specification and requirements documents) and any information needed for developing new software.

Writing reusable code is hard. Not only do developers have to deal with local services, permissions, dependencies and license issues, they also have to provide decent comments explaining exactly *what* their code does and all sorts of documentation that another developer might need to reuse the code properly. The problem with reusability can be that sometimes only the biological results that matter for a given publication comes first, resulting in non-reusable software afterwards that few takes time and effort to make reusable again [15]. It is also, on the other hand, difficult to reuse code because some developers think that it is easier to build something from scratch. In this way, they know exactly *what* is happening, *how* it is happening and *when* it is happening. For some developers, it’s faster to write something again in their own style than to read and understand someone else’s code segment and figure out where to modify changes for the new purpose.

## The Galaxy project

Galaxy is an open, web-based platform for data intensive biomedical research. Whether on the free public server or your own instance, you can perform, reproduce, and share complete analyses.  
galaxyproject.org

The Galaxy framework is a scientific platform with data integration, analysis tools and publishing opportunities that aims to make computational biology accessible to research scientists that do not have a computer programming experience. It is, according to their wiki-page[[1]](#footnote-1), a web-based platform for accessible, reproducible, and transparent computational biomedical research because:

* Users without programming experience can easily specify parameters and run tools and workflows
* Users can repeat and understand complete computational analysis
* Users can share and publish analysis via the web and create Pages, interactive, web-based documents that describe a complete analysis

Galaxy was initially developed for genomics research, but is now used as a general **bioinformatics workflow management system.** It is an open source project implemented using the Python programming language by the Galaxy team and the Galaxy community, which includes users, organizations that install their own instance, Galaxy developers and bioinformatics tool developers. The Galaxy community can use the projects mailing lists, a community wiki, the Galaxy Biostar forum, or the annual meetings to get information or communicate within the community.

### Galaxy objects

Galaxy objects are, in general, anything that can be saved, persisted and shared. Below is a list of galaxy objects that users may encounter:



Figure 2‑9 Galaxy instance home-page  
Green square: Tool-menu  
Yellow square: Workflow  
Red square: Current history  
Blue square: Dataset/history element

#### Histories

Histories are computational analyses with specified input datasets, computational steps and parameters. Histories include all intermediate and output datasets as well. They can easily be labeled, manipulated, and shared/published with anyone, whether they have a Galaxy-account or not.

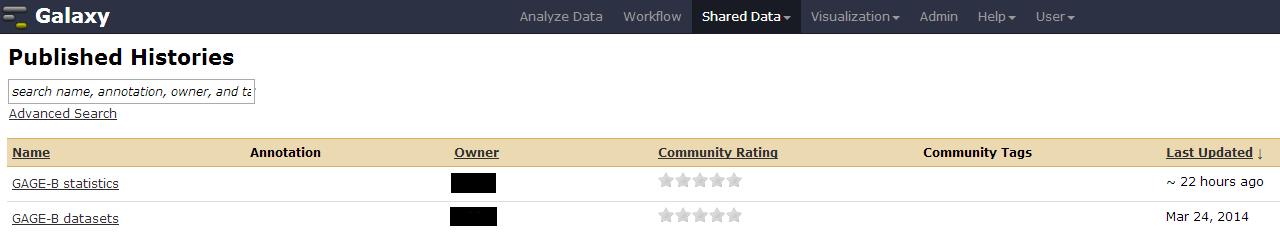


Figure 2‑10 An example of shared or published histories in galaxy

#### Datasets

A dataset is any kind of input or output that is used or produces during each step of an analysis. They can sometimes be referred to as history elements because each dataset is associated with at least one history. The tracking information associated with datasets in a history represent an experimental record of the methods, parameters, and other inputs. These methods are easily extracted into workflows, making an analysis pathway transparent, reproducible, and reusable.

#### Workflows

Workflows are computational analyses that specify all the steps (and parameters) in the analysis, but none of the data. They are used to run the same analysis against multiple sets of input data.

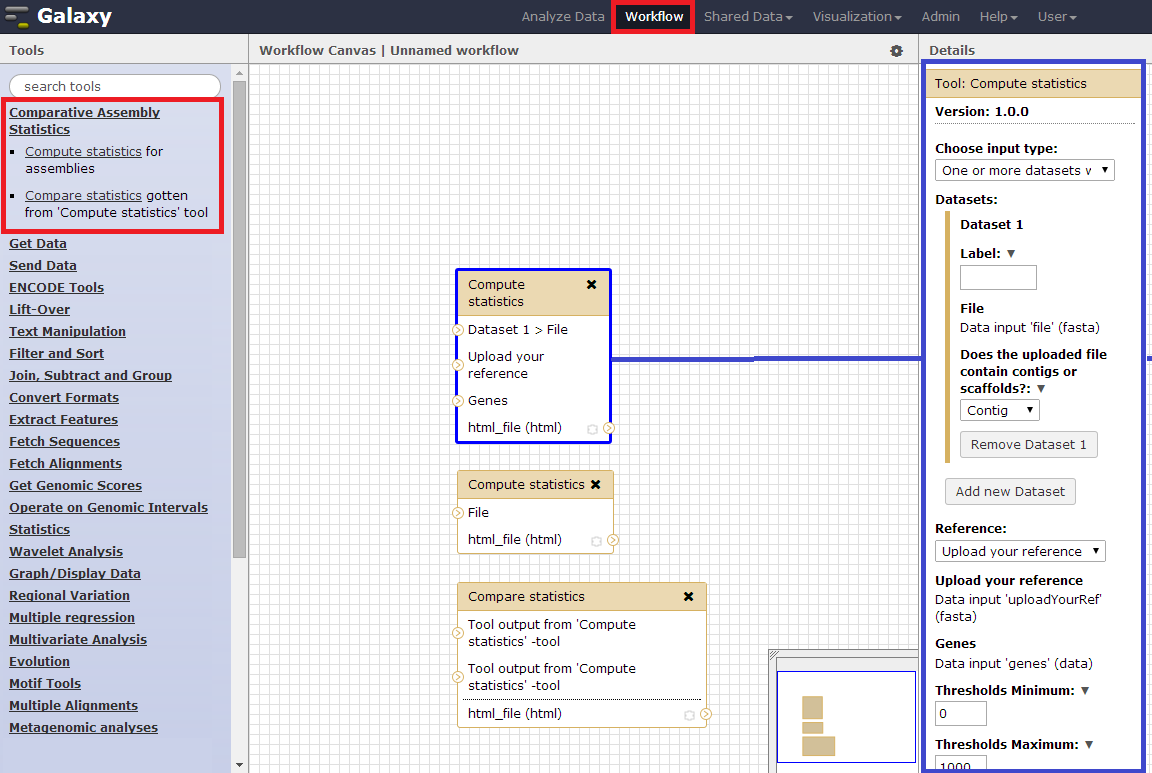


Figure 2‑11 Example of creation of a workflow

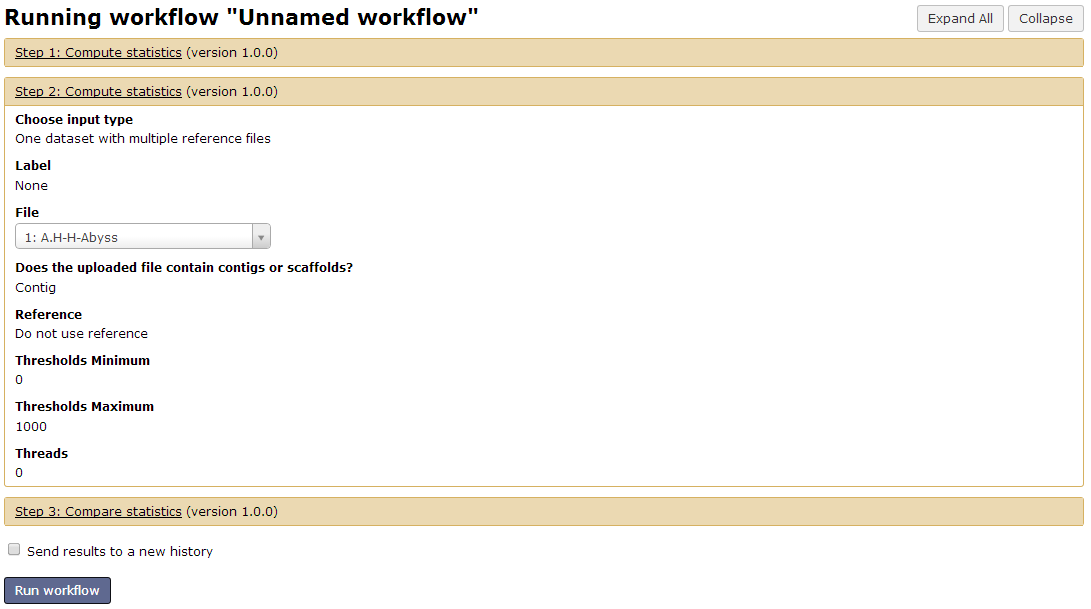


Figure 2‑12 Example of running the workflow from figure 2-6

#### Pages

Histories, workflows and datasets can include user-provided annotation. Galaxy Pages enables the creation of a virtual paper that describes the how and why of the overall experiment. Tight integration of pages with histories, workflows, and datasets supports this goal.

### Toolshed

The Galaxy Tool Shed serves as an appstore to all Galaxy instances worldwide. It is a free service that hosts repositories containing Galaxy tools, managers, datatypes and exported Galaxy workflows. It allows administratros to install freely available Galaxy utilities into their instances while managing external tool dependencies and tool updates, making it easy to share, update and manage tools across all Galaxy instances.

# Materials

This chapter covers all the datasets and software used in our Galaxy tools, and while reproducing the GAGE-B results. All the data used are available at GAGE-Bs webpage or (for assemblies) at http://insilico.hpc.uio.no:24688/history/list\_published. A more detailed explanation about the datasets and software can be found below.

## Datasets

There are three types of datasets used in this thesis, read data, assemblies and reference genomes. The read data were used solely for reproducing the GAGE-B results. The assemblies were used as reference for the reproduced GAGE-B results, and as input for the Galaxy tools. The reference genome were used for both reproducing the GAGE-B results and as parameters for the Galaxy tools.

### Read data

|  |  |  |
| --- | --- | --- |
| Name | Sequencing technology | Size (GB) |
| Aeromonas hydrophila SSU | HiSeq | 7.0 |
| Bacillus cereus VD 118 | HiSeq | 7.0 |
| Bacillus cereus ATCC 10987 | MiSeq | 2.0 |
| Bacteroides fragilis HMW 615 | HiSeq | 7.0 |
| Mycobacterium abscessus 6G-0125-R | HiSeq MiSeq | 2.5 2.0 |
| Rhodobacter sphaeroides 2.4.1 | HiSeq MiSeq | 4.5 1.5 |
| Staphylococcus aureus M0927 | HiSeq | 4.5 |
| Vibrio cholerae CO 1032(5) | HiSeq MiSeq | 2.0 1.5 |
| Xanthomonas axonopodis pv. Manihotis UA 323 | HiSeq | 8.0 |

Table 3‑1 – Species, their sequencing technology and size  
Downloadlink: <http://ccb.jhu.edu/gage_b/datasets/index.html>

All the read datasets used in this thesis are shown above in table 3-1. The reproducing of the GAGE-B papers results was initially started with all datasets in table 3-1. The datasets were first assembled according to the recipe on their site 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. 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.

Sometimes, the raw reads produced by the sequencer are not correct in their whole length because of contaminants, adapter sequences or low-quality sequences. Using the entire read then may introduce artifacts in the genome assembly, and to avoid that, the reads are trimmed or cleaned using various software tools such as for example Trimmomatic [[25](#_ENREF_25)] or, as the GAGE-B researchers have done, by removing adapter sequences and performing q10 quality trimming using the ea-utils package.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ABySS | CABOG | MIRA | MaSuRCA | SGA | SOAPdenovo | SPAdes | Velvet |
| Hiseq | Clean | Clean | Raw | Raw | Clean | Clean | Clean | Clean |
| Miseq | Clean | Raw | Clean | Clean | Clean | Raw | Clean | Clean |

Table 3‑2 – 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 since not all assemblies were carried out successfully. The task was still carried out partially, but not all results are equally relevant or informative. Chapter 5 and 6 will elaborate more on this.

### Assemblies

The final assemblies used in the GAGE-B paper were available online at <http://ccb.jhu.edu/gage_b/genomeAssemblies/index.html> and these were used as input parameters while running the Galaxy tools. Both contig and scaffold files were available for all species and assemblers, except scaffold files for Mira on all species. The assemblies can be accessed from both the GAGE-B site and as history elements in the published histories at http://insilico.hpc.uio.no:24688/u/sabba/p/gage-b-datasets-and-statistics.

### Reference genomes

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. It’s worth noting that in this case, the reference genome is a similar but distinct strain meaning that some differences between the assemblies and the reference genome might be true differences rather than errors.

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 species had quite a list of files available, but 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reference | Type | Size (kB/MB) | | RefSeq |
| 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 3‑3 Reference genome for each dataset  
Source: <http://ccb.jhu.edu/gage_b/datasets/index.html>

## Software

This section covers the various software used in this thesis when it comes to the reproduction of the GAGE-B papers results as well as for the implementation of Galaxy tools. They will cover a short introduction to the software and, when possible, what the software have been used for and which version/release that has been used.

### ABySS

ABySS is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes.  
ABySS webpage[[26](#_ENREF_26)]

The version used in thesis is tha same as in GAGE-B, v1.3.4. This version was released in May 30, 2012 and eliminated two sources of misasseblies, increased the minimum overlap required between two contigs from 30 to 50 and fixed various protability issues.

Many versions have been released since the assemblies were executed for this thesis with version 1.5.1 (released May 08, 2014) being the current release. Any version of ABySS can be downloaded from http://www.bcgsc.ca/platform/bioinfo/software/abyss

### CABOG

CABOG [[27](#_ENREF_27)] is the pipeline revised for 454 data for Celera assembler. This is a de novo whole-genome shotgun (WGS) DNA sequence assembler. Long sequences of genomic DNA is reconstructed from fragmentary data produces by WGS sequencing.

Version used in this thesis are 7.0 and 8.1 (newest release, December 16, 2013) which can be downloaded from http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/

### MIRA

Mimicking Intelligent Read Assembly (MIRA) [[28](#_ENREF_28)] is a multi-pass DNA sequence data assembler/mapper for whole genome and EST/RNASeq projects. It can assemble/map Sanger, 454, Ion Torrent, Solexa (Illumina) and (in development) PacBio reads. The version used in this thesis is 3.4.0 which can be downloaded from <http://www.chevreux.org/project_mira.html>

### MaSuRCA

MaSuRCA (MSRCA) [[29](#_ENREF_29)] is a whole genome assembler that combines the efficiency of the de-Bruijn graph with OLC approaches. It can assemble short Illumina reads or a mixture of short and long reads (Sanger and 454) in projects of all sizes, from bacteria to large plants and mammalian genomes. The versions used in this thesis are 1.8.3 and 2.1.0, while the current release is 2.2.1 (released February 02, 2014). Each release can be downloaded from <ftp://ftp.genome.umd.edu/pub/MaSuRCA/>

### SGA

String Graph Assembler (SGA) is a de novo assembler based on the concept of string graphs that is designed to assemble large genomes from high coverage short reads data. It is very memory efficient because it implements a set of assembly algorithms based on the Ferragina–Manzini index (**FM-index**) that is derived from the **Burrows-Wheeler transform**. [[30](#_ENREF_30)]

The version used in this thesis are 0.9.34 (released August 23, 2012) and 0.10.13 (released January 17, 2014, current release) which can be downloaded from <https://github.com/jts/sga/releases>.

### SOAPdenovo2 + GapCloser

Short Oligonucleotide Analysis Package denovo (SOAPdenovo) is a short-read assembly method (specially for Illumina GA short reads) aimed for assembly of large plant and animal genomes, although it works well on bacteria and fungi genomes as well. It can perform analyses of unexplored genomes and create new opportunities for building a reference sequence.

The newest version, SOAPdenovo2 [[31](#_ENREF_31)] ( released January 28, 2013) has the advantage of reduced memory consumption in graph construction, increased coverage and length in scaffold construction and improved gap closing, to name some.

GapCloser uses the abundant pair relationships of short reads to close gaps that emerges during scaffolding by an assembler.

SOAPdenovo2 version 2.04 (current release) were used in this thesis together with version 1.12 of GapCloser. Both these versions can be downloaded from <http://soap.genomics.org.cn/soapdenovo.html>.

### SPAdes

St. Petersburg genome assembler (SPAdes) is a genome assembler designed for single-and multicelled bacterial data. It works with Ion Torrent, PacBio and Illumina paired-end, mate-pairs and single reads. [[32](#_ENREF_32)]

The current release is version 3.1.0 (released May 29, 2014), but the versions used in this thesis are 2.3.0 (released November 14, 2012) and 2.5.0 (released July 06, 2013) which can be downloaded from http://spades.bioinf.spbau.ru/

### Velvet

Velvet is a de novo genomic assembler that uses de-Bruijn graph to assemble short read data from sequencing technology such as for instance Solexa or 454. It removes errors from reads, produces unique contigs and then retrieves repeated areas between contigs using (when available) paired-end reads and long read information. [[2](#_ENREF_2)]

The versions used in this thesis are 1.2.8 (released November 15, 2012) and 1.2.10 (released October 17, 2013) and can be downloaded from https://www.ebi.ac.uk/~zerbino/velvet/velvet\_[Insert version here].tgz

### Python v2.7

Python is a dynamic, object-oriented programming language that is mainly used as a scripting language, but can also be used for larger applications. It executes at runtime, thus requiring no compilation and combines power with clear syntax. This makes Python code compact and easy to read. It has interfaces to many system calls and libraries, as well as to various windows systems and can be used as an extension language for applications that need a programmable interface. It is also portable, meaning that it can run on various systems including Unix variants, Mac and PCs under MS-DOS and Windows. [[33](#_ENREF_33)]

### Quast v2.2

QUAST is a quality assessment tool for genome assemblies. It evaluates genome assemblies by computing various metrics, including N50, NG50, misassembled or unaligned contigs and genes and operons covered. It also builds plots for different metrics such as cumulative contigs length, all kinds of N-metrics, genes and operons covered, and GC content. [[34](#_ENREF_34)]

### Google charts

Google charts is a simple tool that lets people easily create a chart from some data and embed it in a web page. Currently, line, bar, pie, and radar charts, as well as Venn diagrams, scatter plots, sparklines, maps, google-o-meters, and QR codes are supported. [[35](#_ENREF_35)]

### Sqlite3

This is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. It is used by the tools to access the history id from the database on a given dataset id. [[36](#_ENREF_36)]

### OS module

This module provides a way of using operating system dependent functionality which allows the file to interface with the underlying operating system that Python is running on.[[37](#_ENREF_37)] It is used to create, copy, move and remove files and directories, iterate through a path, check if a path exists, get the content of a directory, join a path, and to validate if a path points to a file or directory.

### Json

Javascript Object Notation (json) is a lightweight data interchange format based on a subset of JavaScript syntax. [[38](#_ENREF_38)] It can be used to load an external json file and to dump the content to a new file.

### Time module

This module provides various time-related functions. [[39](#_ENREF_39)] It is used to get current date and time in a string format.

### Zipfile module

This module provides tools to create, read, write, append, and list a ZIP file. [[40](#_ENREF_40)]

### PyPdf (pdfFileWriter & pdfFileReader)

This is a Pure-Python library built as a PDF toolkit, capable of

* extracting document information (title, author, ...)
* splitting documents page by page
* merging documents page by page
* cropping pages
* merging multiple pages into a single page
* encrypting and decrypting PDF files [[41](#_ENREF_41)]

### Reportlab

This is the ReportLab PDF Toolkit [[42](#_ENREF_42)]. It allows rapid creation of rich PDF documents, and also creation of charts in a variety of bitmap and vector formats.

Canvas from reportlab.pdfgen – The pdfgen package is the lowest level interface for generating PDF documents. A pdfgen program is essentially a sequence of instructions for "painting" a document onto a sequence of pages. The interface object which provides the painting operations is the pdfgen canvas.

defaultPageSize from reportlab.rl\_config – The rl\_config package is used to change the values of several important sitewide properties such as defaultPageSize which is set to A4 as default. [[43](#_ENREF_43)]

### Collections

This module provides alternatives to Python’s built-in containers, dict, list, set, and tuple by implementing specialized container datatypes such as OrderedDict which is a dict subclass that remembers the order entries were added. [[44](#_ENREF_44)]

# Methods

This chapter covers a description of the tool and implementation.

## Reproducing the GAGE-B results

*The recipe for the reproduction of the GAGE-B papers results can be found in Appendix B – Table 1.*

All the runs were performed in a Linux based environment with the programs from the software section under Chapter 3. Each assembly was performed for both MiSeq and HiSeq data were some reads were trimmed/clean and others were raw as described in the GAGE-B papers supplementary file. A common set of data cleaning steps were performed on all datasets since raw sequencing data often contain contaminants, adapter sequences or very low-quality sequences that need to be discarded and the data quality should not dominate the result. The data cleaning were performed by GAGE-B researchers so this thesis took advantages of already trimmed/cleaned sequences. Some assemblies were performed with newer versions of the assembler and both read type and assembler versions are described in Appendix B – Table 2.

## Galaxy

This section will give an overview of the published galaxy-histories with gage-b statistics and cover the methods used for the implementation and testing of the galaxy instance as well as a simple user manual.

### Published Galaxy objects

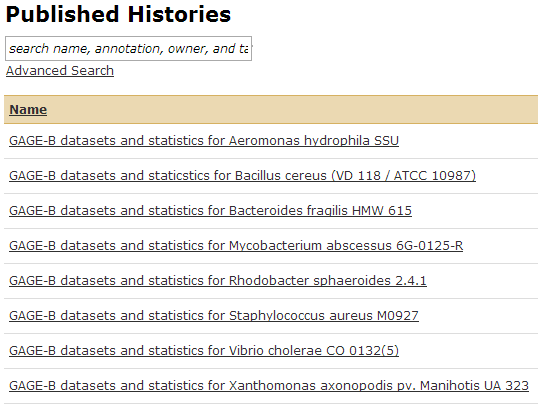


Figure 4‑1 List of published histories

All the GAGE.B assemblies were used as input for several histories that were later published. Each published history covers one species and includes both the datasets and statistics gotten from the tools. Each history consists of species-specific contig and scaffold files for each assembler, except scaffold files for the Mira assembler which were not a part of the downloadable package from GAGE-B. The published histories also contain computed statistics and comparison of statistics where it is possible. The format used for naming the history elements are:

***[A-Z].[A-Z]-[H/M]-[{Assembler}]-[Contig/Scaffold]***This format is used for naming the assemblies based on species, readtype (HiSeq/MiSeq), assembler and datatype (contig/scaffold). All assembler-name were used without version information, except SOAPdenovo2 v2.04 + GapCloser v1.12 were the name was changed to “soap” to increase the readability of the datasets. Examples of this format can be viewed in figure 4-2 element 8-9,23-24. Dataset with this name-format where used as input for the tools to compute statistics.

***Compute statistics on [A-Z].[A-Z]-[H/M]-[Contig/Scaffold]***

The datasets with this name-format contains statistics on a given species HiSeq/Miseq data based on contig or scaffold files. This includes QUAST output with more functionality implemented to the html version of the report, resulting in increased opportunities for visual feedback. See figure 4-2 element 31,32,34 and 35 for example.

***Compute statistics on [A-Z].[A-Z]-[H/M]***

The datasets with this name-format contains statistics on a given species HiSeq/Miseq data based on both contig and scaffold files. This includes QUAST output with more functionality implemented to the html version of the report, resulting in increased opportunities for visual feedback. See figure 4-2 element 33 and 36 for example.

***Compare statistics on [A-Z].[A-Z]-[Contig/Scaffold]\sH+M***  
The dataset with this name-format contains comparison of statistics gotten from the compute statistics tool and merges 2 or more results into one. An example of this is to combine all the contig statistics for a species with both Hiseq and Miseq data into one. Figure 4-2 element 37 and 38 gives an example of this format.

This information is also available on the published page link to page.

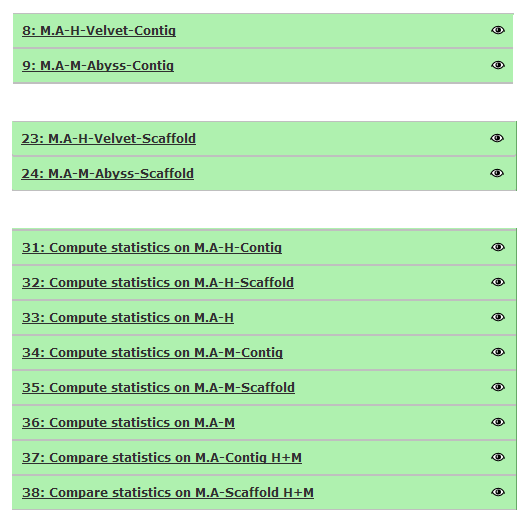


Figure 4‑2 the naming of datasets for Mycobacterium abscessus 6G-0125-R

### Implementation and testing of Galaxy tools

The Galaxy instance used in this thesis is available at link insilico and the code can be downloaded from link github for those more interested in the implementation. The structure of both the Galaxy instance and the tool for this thesis are shown below in Appendix A – Figure 1.

Galaxy’s core functionality is compatible with Python versions 2.6 and 2.7. Two extra modules required by Galaxy (ssl and bz2) are built at the end of the Python compilation process. Make sure that these modules are importable if Python is user compiled. Galaxy requires a few things to run - configuration files, and dependent Python modules called "eggs". However, starting the server for the first time with the command *run.sh* will create/acquire these things as necessary, assuming that Galaxy has internet access to download the eggs. Running this command will also start up the server on localhost and port 8080, so Galaxy can be accessed from a web browser at <http://localhost:8080> allowing developers to run locally without any special environment for running and developing the code. But to access Galaxy over the network, the user has to modify *universe\_wsgi.ini* and change the host setting to 0.0.0.0 to listen on all available network interfaces, or, like in this case, <https://insilico.hpc.uio.no/galaxy_sabbai>. Other settings such as the port to listen to or enabling/disabling live debugging in the browser can also be changed here.

The next Galaxy-file with significance to the development process is *tool\_conf.xml*. This file contains information about which tools to display in the tool menu of Galaxy. The whole menu is enclosed in a toolbox tag, while each “tool header” is defined as a section tag with a unique id (comp\_asm\_stat) used internally, and a name (Comparative Assembly Statistics) displayed externally. This section can hold a variety of tools enclosed in a tool tag which specifies the tools xml-file location. Tool\_conf.xml assume that the path to tool-specific files are saved in a default folder named tools making the tool

<tool file="comparative\_assembly\_statistics/computeStat.xml" />

Actually point to the path tools/comparative\_assembly\_statistics/computeStat.xml

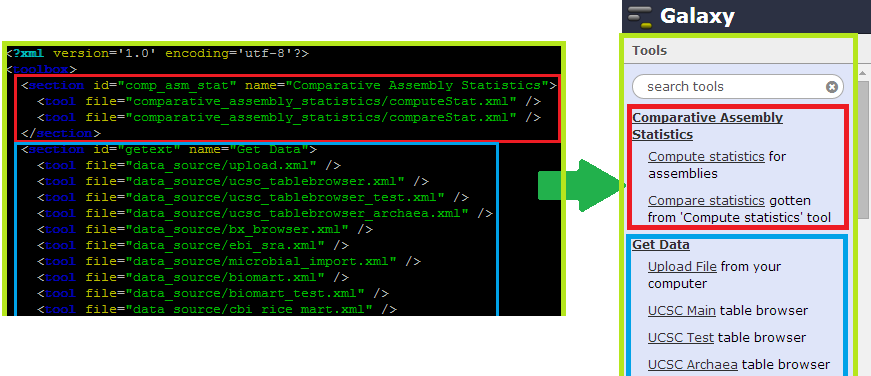


Figure 4‑3 (From left) tool\_conf.xml file and the tool menu in Galaxy

The structure of the tool developed as part of the thesis is inside the tools-folder as another folder named comparative\_assembly\_statistics. This folder contains 7 files, a reference folder containing reference genome an gene annotation used by GAGE-B and quast version 2.2 used to display and run the tools *compute statistics* and *compare statistics.* The tool compute statistics is used to compute statistics as QUAST results with improved visualization features, while compare statistics is used to compare and merge the output gotten from the first tool. The implemented code can be viewed in Appendix A – Figure 2-7. Below is a short description of the 7 files used to run the tools, followed by information about Galaxy’s database folder that might be of interest for a developer.

combineResult.py

This file is used by both tools to combine outputs. It can combine two or more separate outputs gotten from *compute statistics*, or merge the content when *compute statistics* is ran with multiple reference genomes. Running one assembly against multiple reference genomes gives the user one output for each reference and, using this file, a merged view of all the statistics combined.

This file imports the following modules and files to complete its tasks:

* Os, json, time, zipfile, pyPdf, reportlab and collections
* htmlHelpFile.py – This external Python file is used as a help file to generate report.html with advanced visualization features.

computeStat.xml

This xml file is used to create the physical layout of the tool *compute statistics* in Galaxy. It defines the content of input parameters, the output type as html and defines the interpreter and files used in the computation.

computeStat.py

This Python file validates and prepares the tool input/parameters for further use. It runs QUAST on given parameters and alters the report.html file using htmlHelpFile.py and creates separate output folders using combineResult.py if needed.

This file imports the following modules and files to complete its tasks:

* sys, os, and sqlite3
* htmlHelpFile.py – This external Python file is used as a help file to generate report.html with advanced visualization features.
* combineResult.py – This file is used to merge statistics when the input is multiple reference genomes

compareStat.xml

This xml file is used to create the physical layout of the tool *compare statistics* in Galaxy. It defines the content of input parameters, the output type as html and defines the interpreter and files used in the computation.

compareStat.py

This Python file validates and prepares the tool input/parameters for further use. It alters the report.html file using htmlHelpFile.py and uses combineResult.py to merge input data to one single output folder.

This file imports the following modules and files to complete its tasks:

* sys, os, and sqlite3
* htmlHelpFile.py – This external Python file is used as a help file to generate report.html with advanced visualization features.
* combineResult.py – This file is used to merge statistics when the input is multiple reference genomes

draw\_cumulative\_plot.js

This is a javascript file that contains to functions that take advantage of google charts to create interactive plots used in report.html file in the output of both tools. A third function is used to toggle image, i.e. hiding the plot at the users command when displayed. The first function creates bar charts and the second funtion creates scatterplots. The file itself is copied to a script folder in the output folder, enabling the use of the functions in the report.html file.

htmlHelpFile.py

This is a Python file included by combineResult.py, computeStat.py and compareStat.py and contains the code to dynamically create dropdownlist from which the user can choose to make a plot from. It also contains the script code needed to import draw\_custom:plot.js into report.html and to place visualization features such as dropdownlists, create plot-buttons and a div to display the plots including a hide-plot-option. This file imports the json package to get information needed to create dropdown lists for the report.html file.

The last folder that is worth mentioning is the database folder. This folder contains a couple of folders and files including *universe.sqlite* which is the database containing anything related to Galaxy worth saving. The folders that are interesting for this thesis is the *job\_working directory* which holds all temporary files during an execution of a tool and *files* folder that contains the history elements and any output files/folders beside whats stored in each history element. One or more folder named with numeric values incremented by 1 for each new folder, starting with 000, is created inside *files.* Each folder can store a fixed number of elements before a new folder (e.g 001) is created to store the next elements. These folders store history elements as dataset\_x.dat and any other file in a folder named dataset\_x\_files where x is a a unique, numeric value which is incremented by 1 for each entry. This is worth nothing because if a user deletes a history-element, that element will be hidden from the history, but not deleted from disk and still accessible from the *files* folder or by undeleting the history element*.*

### Simple user manual for the tools

There are a total of 2 tools that form the basis of this thesis:

*Compute statistics* which uses at least one assembly (reference genome and genefile is optional) to compute statistics as quast results with improved visualization features. And *Compare statistics* which uses at least two outputs from the first tool to compare and merge the outputs into one single output.

#### Content

Compute statistics – for assemblies

Get input data

Add tool-input and parameters

Execution

View results

Compare statistics – gotten from the “Compute statistics” tool

Get input data

Add toll-input

Execution

View results

This section will provide a simple user manual for each tool, including step-by-step instructions using one importable, published history (test-data1) as shown in figure 4-3.

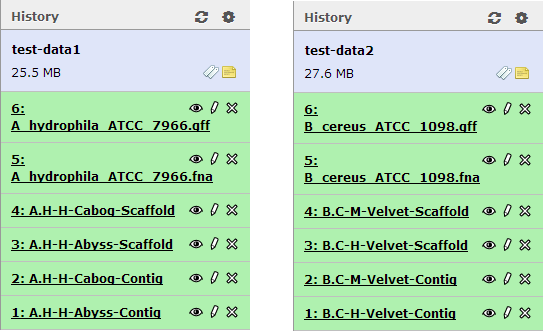


Figure 4‑4 One history with test-data used in the user manual

#### Compute statistics

This tool uses at least one assembly (reference genome and genefile is optional) to compute statistics as quast results with improved visualization features. The output from this tool can be used as input for “Compare statistics”-tool where 2 or more results can be merged into one.

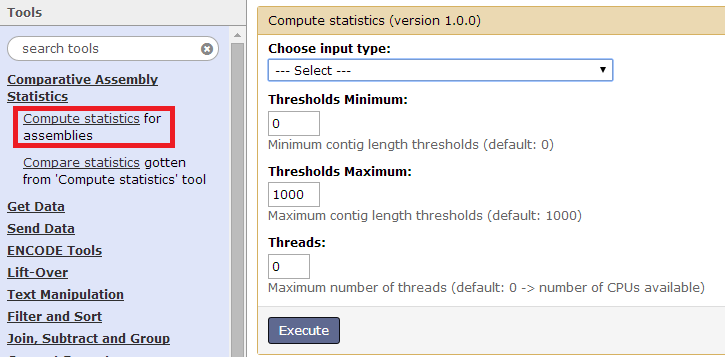


Figure 4‑5 Default startpage for the «Compute statistics»-tool

Get input data

There are three different types of input data for this tool that you can upload yourself:

Assembly file (mandatory to run the tool)

Can contain contigs or scaffolds

Reference file (Optional)

Containing at least one reference genome

Gene file (Optional)

Containing genes from at least one genome

The tool “Upload File” is used to upload any kind of the input data mentioned above. This tool offers a number of options as to how a user can upload data and information about the data such as file type and genome. A screenshot of the tool can be seen in Figure 4-5.

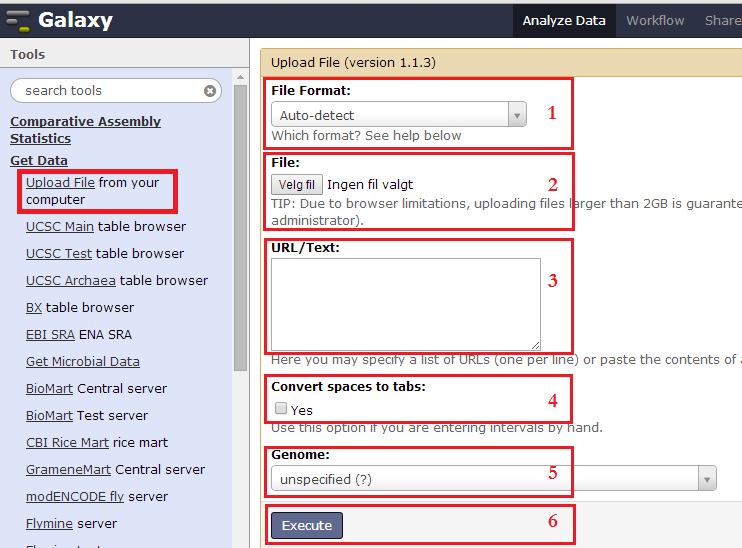


Figure 4‑6 Blankpage for the tool «Upload file»  
1. File format – Let the tool auto-detect file format or choose a format (e.g fasta or bam) from list.  
2. File – Upload an existing file from computer  
3. URL/Text – Write the URL to a file or write the content directly in the textbox  
4. Convert spaces to tabs – An option used to mark the content in 3 as tab separated file  
5. Genome – Optional to specify which genome the file correspond to  
6. Execute – Press this to start the file upload to current history

Add tool-input and parameters

The first choice to make is to choose one of the two options in the input type dropdown list:

1. One or more datasets with at most one reference

This option allows the user to compare several assemblies with each other, with or without a reference genome depending on how much details the user want. To add an assembly to the tool, press the button “Add new Dataset”. Figure 4-6 contains a screenshot of the new fields that appear upon pressing the button. The user can add a label that will be used in the final report, choose assembly from historyelements in current history and set the file type as contigs or scaffold. Press the button “Add new Dataset” to add more assemblies or “Remove Dataset 1” to remove dataset 1.

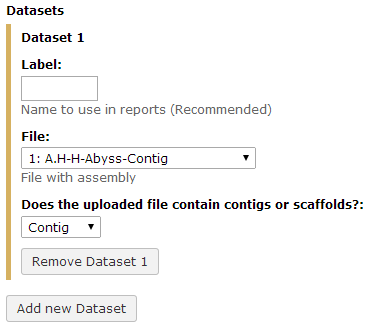


Figure 4‑7 Screenshot of new fields available after pressing “Add new dataset”

The user can compute statistics for at least one assembly with or without a reference genome. Whether or not a reference genome is used is set by the option “reference” where the user can choose to:

Not use a reference genome at all. This option will provide the user with basic statistics such as number of contigs, total length of bases in assembly, N50 and GC-content.

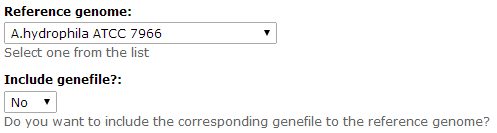
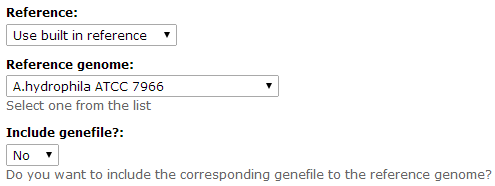
Use one built in reference genome with optional use of corresponding gene file

Figure 4‑8 Screenshot of the option “Use built in reference”

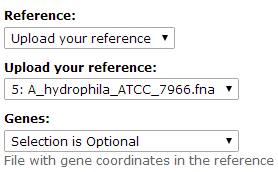
Upload another reference genome with optional upload of corresponding gene file. This option allows the user to upload one reference file containing a single reference genome. 

Figure 4‑9 Screenshot of the option «Upload your reference»

2. One dataset with multiple reference files

This option allows the user to compare one assembly with multiple reference files which can be used to for instance determine which strand of a genome is the most accurate to a reference genome on a given assembly. To add an assembly to the tool, press the button “Add new Dataset”. Figure 4-6 contains a screenshot of the new fields that appear upon pressing the button. The user can add a label that will be used in the final report, choose assembly from historyelements in current history and set the file type as contigs or scaffold. Press the button “Add new Dataset” to add more assemblies or “Remove Dataset 1” to remove dataset 1.

The user can compute statistics for at least one assembly with or without a reference genome. Whether or not a reference genome is used is set by the option “reference” where the user can choose to:

Not use a reference genome at all. This option will provide the user with basic statistics such as number of contigs, total length of bases in assembly, N50 and GC-content.

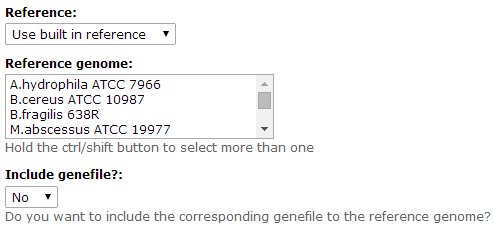
Use at least one built in reference genome with optional use of corresponding gene file 

Figure 4‑10 Screenshot of the option «Use built in reference”

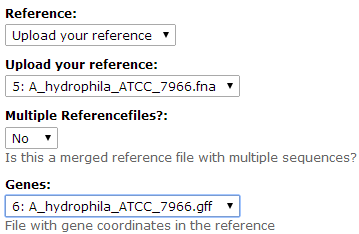
Upload another reference genome with optional upload of corresponding gene file. This option allows the user to upload one reference file containing either a single sequence or multiple sequences merged into one file. 

Figure 4‑11 Screenshot of the option «Upload your reference»

Last, but not least, some quast specific parameters such as minimum/maximum thresholds and the number of threads can be customized regardless of the option chosen in “Choose input type”.

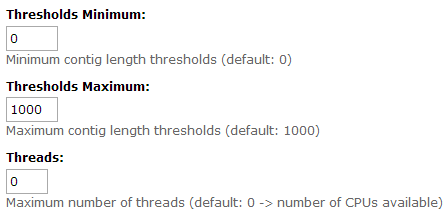
The minimum and maximum threshold parameters are used in # contigs ≥ x and total length (≥ x) metrics and the default value is [0,1000].

Figure 4‑12 Screenshot of quast specific parameters with their default value

The threads parameter allows the user to set the maximum number of threads used in the execution of the tool. If the tool fails to determine the number of CPUs, the number is set to 4. The default value is the number of CPUs available.

Execution

A history element with the output is created upon execution and the color of the history element determines which part of the execution the tool is in. It will first appear in the current history as queued (gray), then running (yellow) and finally done successfully (green) or with problems encountered (red). At this point, clicking on the name of the history element will show information about the tool output. The history element can be saved by clicking on . The user can get detailed information about the execution by clicking on  and the tool can be rerun with the same input by clicking on . Clicking on the pencil icon  will allow the user to edit any attributes related to this history element and this history element can be deleted by clicking on .

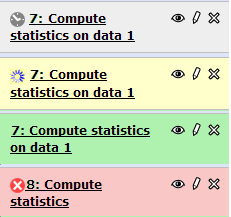


Figure 4‑13 Stages of tool execution

View results

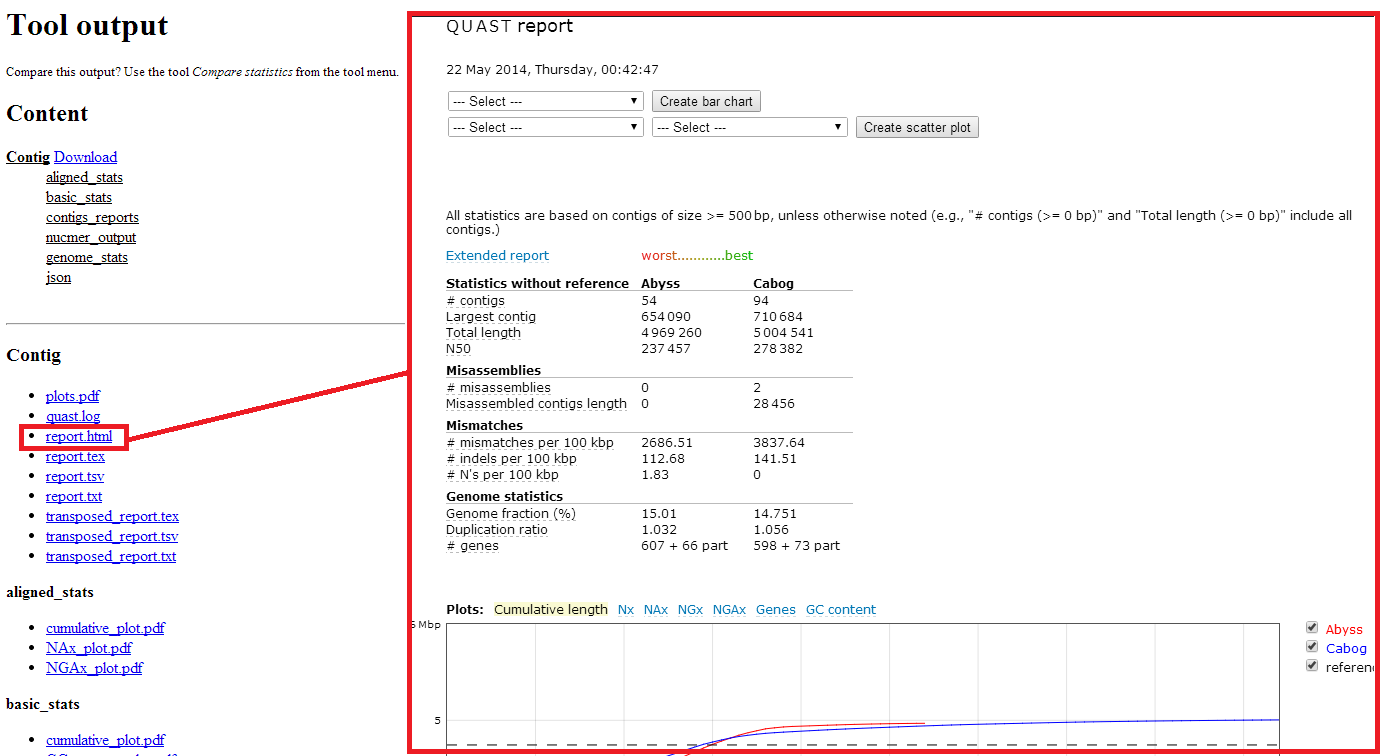


Figure 4‑14 Example of tool output and in browser view of report.html

The user can view a list of all statistics and corresponding files (plots, json-files, contigs\_reports etc.) by clicking on the eye icon  of the history element. Clicking on one of these files will show the file in browser when possible or automatically download it. All statistics can also be downloaded as a single zip-file for further external investigation. Although all the computed files are available, some files are more informative than other and most users may only be interested in these files. The next subsection gives a short introduction to the most used files.

report.html

The most informative file is *report.html*. This file displays statistics, provides visualization features and is highly interactive. The statistics are displayed in a table manner where the columns can be dragged and repositioned according to the users need and requirements. An example of this is shown in Figure 4-13. The user can also hover over a metric to get instant information about the metric or hover over a numeric value to get each assembly’s performance on that given metric. The color of a row’s numeric data will change from black to a gradual transition from green to red where green is the best result and red is the worst result if the user hovers over a numeric value in the table.

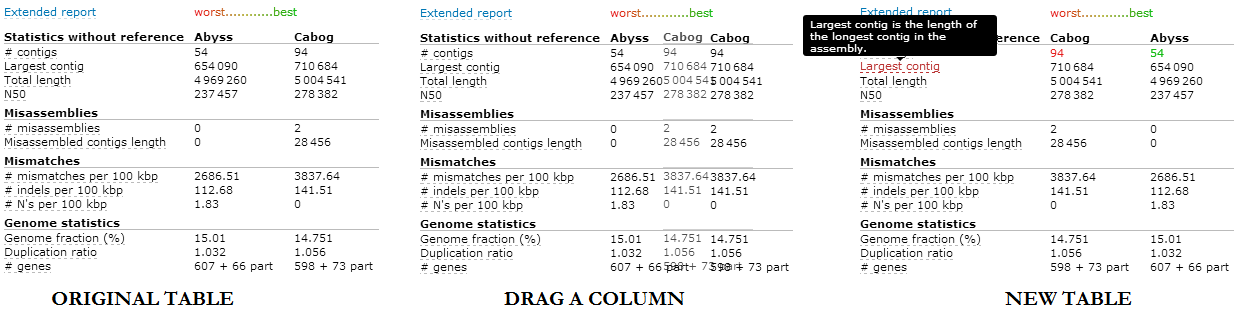


Figure 4‑15 eXAMPLE OF INTERACTIVE TABLE IN REPORT HTML  
nEW TABLE SHOWS INFORMATION YOU GET BY HOVERING OVER A METRIC AND INSTANT EVALUATION OF ASSEMBLY-PERFORMANCE ON A GIVEN METRIC FROM HOVERING OVER A NUMERIC VALUE

This file provide the user with at least 3 (cumulative length, Nx, GC content) and up to 10 predrawn, interactive plots depending on the statistics available. These plots are interactive by giving users information about a graph if they hover over it and allowing them to add or remove an assembly from the plot by clicking on the checkboxes in the legends (as shown in Figure 4-14).

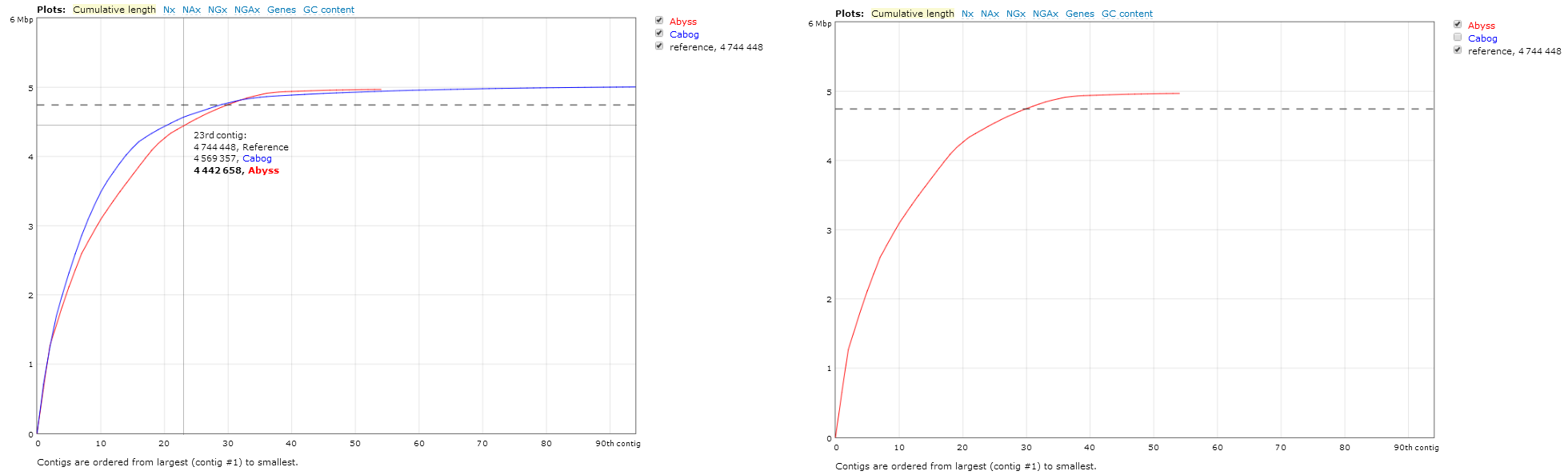


Figure 4‑16 Predrawn plot in report.html  
Left: Original plot (cumulative length) with mouse hovering over the graph  
Right: Same plot, but CABOG is unchecked from the legend at the rigth side of the plot making a plot with only ABySS data

Each of these predrawn plots are mentioned below with a short description gotten from QUAST’s user manual [26]:

|  |  |  |
| --- | --- | --- |
| Plot | Description | Further information |
| Contig alignment | Shows alignment of contigs to the reference genome and the positions of misassemblies in these contigs. Contigs that align correctly are colored blue if the boundaries agree (within 2 kbp on each side, contigs are larger than 10 kbp) in at least half of the assemblies, and green otherwise. Blocks of misassembled contigs are colored orange if the boundaries agree in at least half of the assemblies, and red otherwise. Contigs are staggered vertically and are shown in different shades of their color in order to distinguish the separate contigs, including small ones. If the reference file consists of several sequences all of them are drawn on the single plot horizontally next to each other. |  |
| Cumulative length | Shows the growth of contig lengths. On the x-axis, contigs are ordered from the largest to smallest. The y-axis gives the size of the x largest contigs in the assembly |  |
| Nx | Shows Nx values as x varies from 0 to 100 %. | Nx is the length for which the collection of all contigs of that length or longer covers at least half an assembly |
| NGx | Shows NGx values as x varies from 0 to 100 % | NGx is the length for which the collection of all contigs of that length or longer covers at least half a reference genome. This metric is computed only if a reference genome is provided |
| GC content | Shows the distribution of GC content in the contigs. The x value is the GC percentage (0 to 100 %). The y value is the number of non-overlapping 100 bp windows which GC content equals x % | GC content is the total number of G and C nucleotides in the assembly, divided by the total length of the assembly |
| Cumulative length for aligned contigs | Shows the growth of lengths of aligned blocks. If a contig has a misassembly, QUAST breaks it into smaller pieces called aligned blocks. On the x-axis, blocks are ordered from the largest to smallest. The y-axis gives the size of the x largest aligned blocks. This plot is created only if a reference genome is provided |  |
| NAx | Similar to Nx but for the NAx metric and is created only if a reference genome is provided | NAx is similar to Nx, but in this case aligned blocks instead of contigs are considered.  Aligned blocks are obtained by breaking contigs in misassembly events and removing all analigned bases. |
| NGAx | Similar to NGx but for the NGAx metric and is created only if a reference genome is provided | NGAx is similar to NGx, but in this case aligned blocks instead of contigs are considered.  Aligned blocks are obtained by breaking contigs in misassembly events and removing all analigned bases. |
| Genes | Shows the growth rate of full genes in assemblies. The y-axis is the number of full genes in the assembly, and the x-axis is the number of contigs in the assembly (from the largest one to the smallest one).  This plot could be created only if a reference and genes annotations files are given |  |
| Operons | Is similar to the genes plot but for operons |  |

Table 4‑1 Description of predrawn plots in report.html

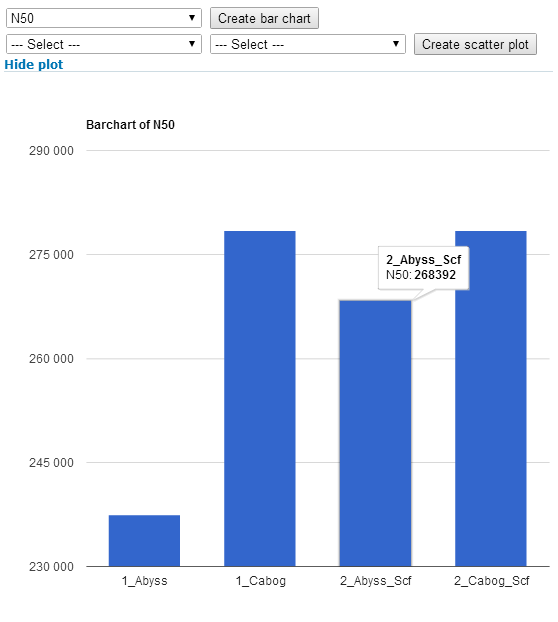
The user can also create other plots with metrics of their own choice as either a bar chart for one metric or as a scatterplot for two metrics. These plots are extra, interactive, visualization features implemented on top of QUAST’s original features for this file (report.html).

Figure ‑ Bar chart in report.html

The bar chart is dynamically created by choosing a metric from the first dropdown-lists and clicking “Create bar chart”. It is displayed right beneath the dropdown-lists for scatterplot. A “Hide plot” option is also available upon creation, making it easier to hide the bar chart if the users do not want to display it anymore. The bar chart is can give you information about the bar-name, metric and numeric value when hovering over a bar.

The scatterplot on the other hand is created by choosing metrics from the two adjacent dropdown-lists and pressing “Create scatter plot”. The plot is displayed right beneath the adjacent dropdown-lists and can, just like bar chart, be hidden by clicking on the blue *Hide plot*. These plots come with a bit more functionality than the bar charts as they give the user opportunity to add/remove assemblies from the plot by clicking on the legend. Even though there is no checkboxes, which might confuse some users, clicking on a name or box in the legend will automatically display or hide that point in the plot. This functionality exists in the predrawn plots as well but the difference from them is that the scatterplots rescale to fit the new number of elements displayed (Figure 4-17-Right plot). The user can hover over a point or over a legend to display horizontal and vertical axes as well as assembly-name, x-value and y-value of that given point or assembly. Clicking on a point marks the horizontal and vertical axes of that point allowing the user to hover over another point to see the difference visually (Figure 4-17-Left plot). Note that only one point at a time can be selected and that selecting a point automatically deselect any previously selected point.

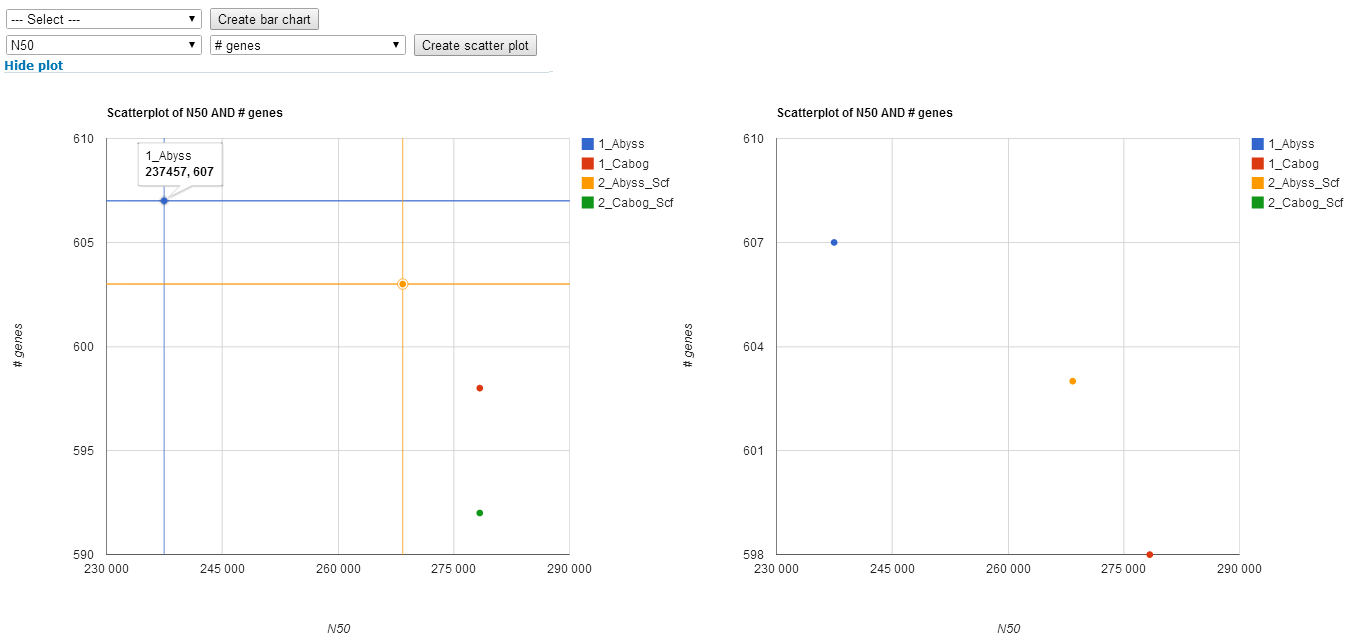


Figure 4‑18 Scatterplot in report.html   
left: Original plot showing selectable point (yellow) with a hovered point (blue) rigth: 2\_Cabog\_Scf removed from displayed elements

Plots.pdf

Plots.pfd is a file that contains all the dynamic, predrawn plots from report.html as static images.

Report.xxx

There are three other types of report files beside report.html that can be of some use. What they all have in common is that they contain an assessment summary and that the transposed version of the files are named with transposed\_ as prefix.

Report.txt contains the summary in a simple text format

Report.tsv contains a tab-separated version of the summary, suitable for spreadsheets

Report.tex contains a LaTeX version of the summary

#### Compare statistics

This tool compares and merges at least two tool outputs (statistics) gotten from the “Compute statistics”-tool.

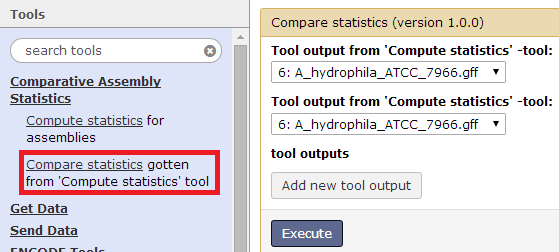


Figure 4‑19 Default startpage for the «compare statistics» tool

Get input data

Input data for this tool is obtained by running “Compute statistic”-tool and use the output of that tool. This tool will automatically detect all files and folders that the output points to and merge them properly.

Add tool-input

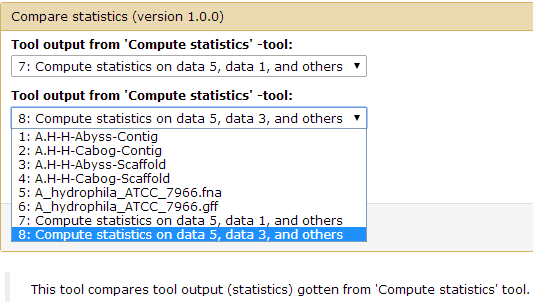


Figure 4‑20 Screenshot of how to choose input data for “Compare statistic”

This tool requires that you choose input from dropdown lists with the history elements from the current history. Note that multiple inputs with the same history element will be omitted and that this tool requires at least two unique history elements.

Execution

A history element with the output is created upon execution and the color of the history element determines which part of the execution the tool is in. It will first appear in the current history as queued (gray), then running (yellow) and finally done successfully (green) or with problems encountered (red). At this point, clicking on the name of the history element will show information about the tool output. The history element can be saved by clicking on . The user can get detailed information about the execution by clicking on  and the tool can be rerun with the same input by clicking on . Clicking on the pencil icon  will allow the user to edit any attributes related to this history element and this history element can be deleted by clicking on .

View results

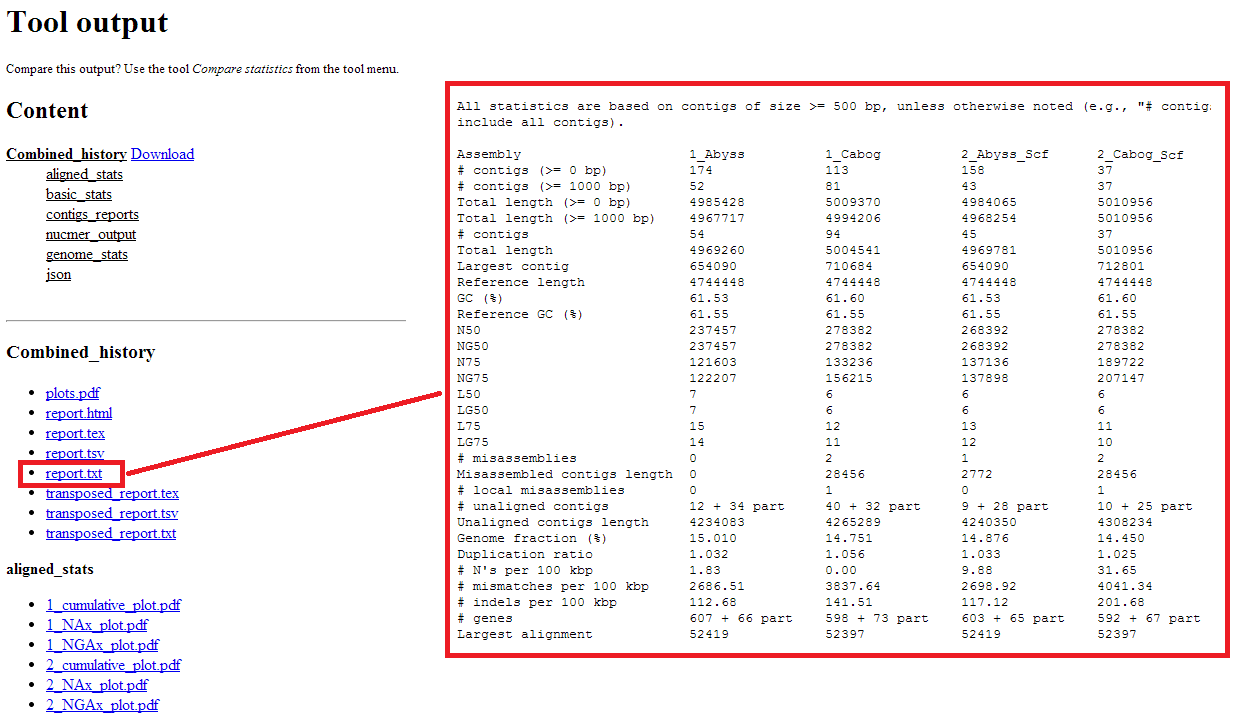


Figure 4‑21 Example of tool output and in browser view of report.txt

The user can view a list of all statistics and corresponding files (plots, json-files, contigs\_reports etc.) by clicking on the eye icon of the history element. A number is added to distinguish between each input elements unique files and tell which column in the report files correspond to which input element. Clicking on one of the files will show the file in browser when possible or automatically download it. All statistics can also be downloaded as a single zip-file for further external investigation. For further details on the output look at the *view result* section of *Compute statistics*

# Results

The results computed for this thesis is written below into subsections covering whether it is contradicting GAGE-B results, reproduction of GAGE-B results or reuseability of datasets. Appendix B- Table X-Y presents various detailed statistics on each assemblies (obtained from QUAST), while this chapter will focus on the following metrics:

* The number of contigs (≥ 200 bp) or scaffolds (≥ 500 bp)
* N50 statistics
* Corrected N50 (NA50) obtained after splitting contigs/scaffolds at each error
* The number of relocations, translocations and inversions (# misassembly) affecting at least 1000 bp, which is determined by comparison to the reference genome
* The number of local errors such as misjoins (# local misassembly) where the left and right pieces map onto the reference genome to distinct locations that are <1000 bp apart, or that overlap by <1000 bp
* Number of fully unaligned contigs that have no alignment to the reference sequence
* The fraction of reference genome covered (Genome fraction %)
* The amount of overlaps among contigs/scaffolds that should have been merged. Failure to merge overlaps leads to overestimation of the genome size and can create two copies of sequences that exist in just one copy. (Duplication ratio).
* The number of complete genes in the assembly (# genes)

All the metrics above are used as assessment of assemblies in GAGE-B, except for the number of genes which replaces GAGE-B-metric *the number of proteins fully contained in contigs.* The protein-metric used in GAGE-B has been excluded because it is retrieved from an external source other than QUAST (tblastn) and this thesis is focusing on QUAST-related statistics only. The number of genes in the assembly was chosen as a replacement because it gives more relevant information compared to other QUAST metrics when considering the the overall metric collection written above.

Both MiSeq and Hiseq data have been used as input for each assembler and similarity or variance of the results is stated below.

## inconsistent GAGE-B results

There are quite a few contradicting results in the GAGE-B papers supplementary material versus statistics gotten by running QUAST on GAGE-B supplied assemblies. First of all, almost all misassemblies whether they are local og global are different.

## Reproducing GAGE-B results

All GAGE-B results are reproduced using GAGE-B supplied reads and recipe for assemblies unless otherwise noted in Appendix B –Table X. The results from assemblies performed on *Vibrio cholerae* are described below, each separated by the perforing assembler.

### Assembler specific comparison to GAGE-B results

Abyss

Cabog

Mira

Msrca

Sga

The results for SOAPdenovo2 version 2.04 with GapCloser 1.2.12 are more of a contradiction to each other. While the number of contigs and genes, N50 and NA50 for both Miseq and Hiseq data (contig) give poor results compared to GAGE-B, the number of local and global misassemblies are quite low in comparison. Scaffold files on the other hand perform almost as good as the GAGE-B results except for the number of misassemblies were the global number gives 8 fewer misassemblies and the local gives 34 more misassemblies.

The Hiseq datasets for both contig and scaffold in SPAdes 2.3.0 performs better overall compared to GAGE-B results, except for 7 more misassemblies (contig) and 2 more unaligned contigs (scaffold). Miseq data on the other hand performs equally good as the expected outcome from GAGE-B results.

Velvet 1.2.08 runs performs better when counting the number of contigs, but has in return more scaffolds on both Miseq and Hiseq data compared to GAGE-B results. The N50 values were the opposite with overall worse values for contigs and better values for scaffolds compared to GAGE-B. The number of local misassemblies, unaligned contigs, genes and genome fraction for contig files are quite similar to the original GAGE-B results. While most values for scaffold files are quite similar to the GAGE-B results, scaffold files perform better in regards to the number of both local and global misassemblies (exept # misassemblies for Hiseq data) compared to GAGE-B results.

### Comparison of assemblies

## Reuseability of GAGE-B data

This section describes results gotten on the same dataset as in section 5.2, but with newer versions of assembler where possible. The results are separated into subsections based on the performing assembler.

### ABySS

Avvik eller ikke og hvor stort det er.

### CABOG

### MIRA

### MaSuRCA

### SGA

### SOAPdenovo

### SPAdes

### Velvet

Velvet got an highe

### Comparison of assemblies

# Discussion

## Interpreting the GAGE-B results

## Analysis of galaxy tool

### Performance

The Galaxy tools work for assemblies with both scaffold- and contig-files and can handle an arbitrary number of assemblies and reference genomes. The most computational intensive part of the tool *Compute statistics* is the assessment of the assemblies which is performed by QUAST. The more assemblies selected as input the longer time would one expect that the run will use. The tool *Compare statistics* is in comparison quite fast as it only merges already computed statistics and therefore does not perform any computational intensive tasks.

### Potential use

The Galaxy tools could be used as supplement to other assembly evaluation tools. It performs better in terms of visual features compared to existing statistical applications such as QUAST and provides users with a simple, graphical user interface in a controlled, scientific environment dedicated to people with less or no computer programming experience. The frameworks reusability of previous runs makes it ideal for researchers as the results gotten from the tools, easily can be reproduced with the same parameters and input, assuming that the input files are published or downloadable from a reliable source.

### Weakness

The major weaknesses to the new Galaxy tools are when it comes to the new visualization features. When it comes to scatterplot, it should be more intuitivt so that the user know that it is possible to check and uncheck which assemblies to display in the plot. As it is now, this feature is ambigious and need to be explained explicitally in the user manual. The size of the plots can also be considered a weakness, as it is set to a default size now and may be too big for some metrics or number of assemblies.

Another weakness is that when users run one of the tools, viewing the results directs them to a list of all files available instead of redirecting then right away to the most used file (report.html). This can be viewed a weakness because new users might get confused when presented with so many files at once. But at the same time, considering the number of report.html files that can be present in one single output makes it difficult to choose which report.html file the tool should automatically redirect to.

A lately discovered weakness is with regards to merging results with different thresholds values. For instance if the input for threshold is [200,1000] for one run and [500,1000] for another run, then the tool *compare stat* cannot complete it’s task.

## Further work

There are still some things that can be considered for further work. When it comes to the new visualization features, maybe making the size of the plots user defined can be one thing to be considered by the next generation of developers. Another feature one might add to the plots is making scatterplot for one metric to look at the differences and similarities based on groups of assemblies. An example of this is looking at the N50 metric for Vibrio cholera where group 1 (x-value) is assemblies from MiSeq data on multiple assemblers and group 2 (y-value) is assemblies from HiSeq data on multiple assemblers. Having a scatterplot in this way can give the user visualized information about a group’s overall performance against another group to determine if for instance the MiSeq data performs better on all assemblers than HiSeq data on a given metric. The next thing that could be implemented to the plot-features are legends with checkboxes, allowing users an intuitive way of excluding or adding some assemblies from the plots, or maybe make a selectable option to choose which assemblies to create the plot upon.

Automatically redirect to the most used and informative file (report.html) instead of showing a list of files available can be an option to be revised for further development. As mentioned in *6.2.3 Weaknesses* it can be difficult to choose which report.html file to redirect to if it is many folders in one output, so maybe choosing this as input parameters for the tool can be an option.

Cross use of tool outputs for the plots or selecting some assemblies from different outputs can also be considered an option for further development. This can come in handy if a user has 2 outputs, but only need to merge one assembly from the first output and three assemblies from the second. It could be an option to choose assemblies from outputs instead of merging all the assemblies in both outputs or rerun a tool with the four assemblies needed since thay have all been run once.

The tools can, as for now, be accessed from the university’s server or be downloaded from a github repository, but the most ideal solution would probably be to export the tools to Galaxy’s toolshed. As explained earlier in Chapter 2.4.3 Toolshed, uploading the tool to toolshed will make it easier to share, install and use the tools on more than one Galaxy instance.

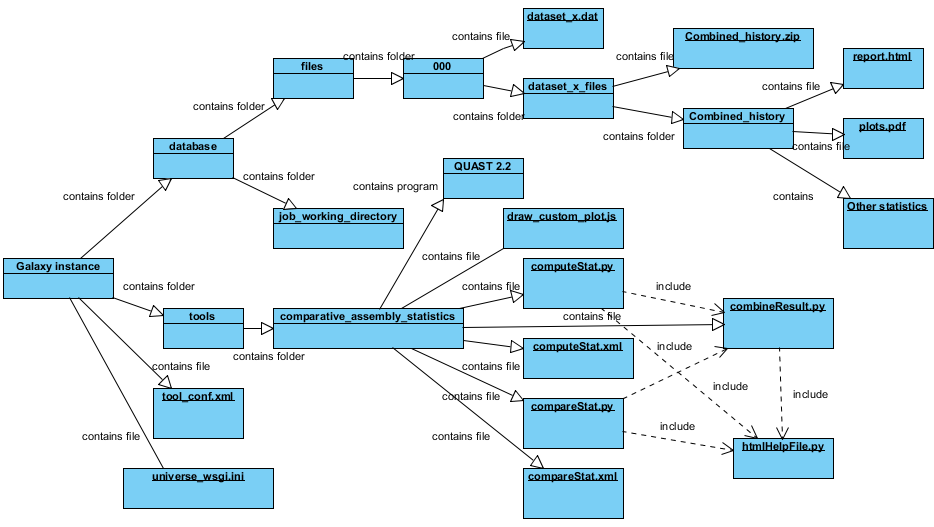
A new version of QUAST (version 2.3) has been released on January 17, 2014 and an upgrade from QUAST 2.2 in the Galaxy tools should be considered. The new version (current release) of QUAST comes with contig alignment plots, updated misassemblies detection logic, full report in PDF format, and many other features which can make an upgrade in the Galaxy tools highly desireable.

## Conclusion

Glossary

|  |  |
| --- | --- |
| Amplicon | A piece of DNA or RNA that is the source and/or product of natural or artificial amplification or replication events |
| 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 |
| bioinformatics workflow management system | A specialized form of workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or a workflow, that relate to bioinformatics |
| bridge PCR | Amplification where fragments are amplified upon primers attached to a solid surface and form "DNA colonies" or "DNA clusters" |
| Burrows-Wheeler transform | An algorithm used in data compression techniques where the transformation is done by sorting all rotations (permutations) of a text in lexicographic order, then taking the last column only. |
| 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 | To sequence uncharacterized genomes where there is no reference sequence available, or known genomes where significant structural variation is expected |
| 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* |
| emPCR | A clonal amplification performed in an oil-aqueous emulsion |
| eukaryote | An organism whose cells contain membraine-bound organelles and whose DNA is enclosed in a cell nucleus and is associated with proteins |
| exome | All DNA that is transcribed into mature RNA in cells of any type. Is a part of the genome formed by *exons* |
| exons | The DNA sequence within a gene and the corresponding sequence in RNA transcripts |
| FM-Index | a compressed full-text substring index based on the *Burrows-Wheeler transform* |
| fungus | About 80,000 known species of organisms of the kingdom Fungi, which includes the yeasts, rusts, smuts, mildews, molds, mushrooms, and toadstools |
| galaxy-announce | Galaxy mailing list used for announcements of interest to the Galaxy community |
| galaxy-dev | Galaxy mailing list used for local installation, configuration, and tool integration help, or to propose new features. |
| genome | The complete complement of an organism's genes; an organism's genetic material |
| hiSeq sequencer | The main workhorse, most expensive with the highest output; best choice for a large number of samples or if you need a lot of reads per sample |
| in vitro | Studies that are performed with cells or biological molecules outside their normal biological context; for example proteins that are examined in solution, or cells in artificial culture medium |
| in vivo | Studies where the effects of various biological entities are tested on whole, living organisms |
| indel | An insertion or the deletion of bases in the DNA |
| inversion | A misjoin of a scaffold/contig where the two pieces map to the opposite strands on the same chromosome |
| miSeq sequencer | The desktop instrument with quick and inexpensive runs; best choice for smaller number of samples and if you need quick turnaround times |
| mRNA | Abbreviation for *messenger-RNA* |
| messenger-RNA | RNA molecules that convey genetic information from DNA to the ribosome, where they specify the amino acid sequence of the protein products of gene expression |
| non-coding RNA | A functional RNA molecule that is not translated into a protein |
| nucleus | The chromosome-containing organelle of a eukaryotic cell |
| paired end reads | Reads that are sequenced from both ends and referred to as R1 and R2. Usually there is a "gap" in between them and although we don't know the sequence of DNA in between R1 and R2, we still have gained useful information from the knowledge that R1 and R2 are next to each other with a known orientation and distance apart |
| PCR | Abbreviation for *polymerase chain reaction* |
| phage | A virus that infects bacteria; also called a bacteriophage |
| polymerase chain reaction | A biochemical technology in molecular biology used to amplify a single or a few copies of a piece of DNA across several orders of magnitude, generating thousands to millions of copies of a particular DNA sequence |
| polymorphism | To have many forms, in our case with repeats, when there are small differences in the repeats based on for instance the length of the repeat |
| protein | A three-dimensional biological polymer constructed from a set of 20 different monomers called amino acids |
| protein synthesis | A process in which cellular *ribosomes* create *proteins* |
| 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 |
| Ribosomal-RNA | The RNA component of the *ribosome*, essential for protein synthesis in all living organisms |
| ribosome | A large and complex molecular machine, found within all living cells and serves as the primary site of biological *protein synthesis* by linking amino acids together in the order specified by *mRNA* molecules |
| ribonucleic acid | A family of large biological molecules that perform multiple vital roles in the coding, decoding, regulation, and expression of genes |
| RNA | Abbrevation for *Ribonucleic acid* |
| rRNA | Abbrevation for *ribosomal-RNA* |
| 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 |
| single end reads | As opposed to *paired end reads,* single end reads are only sequenced from one end of the fragment |
| template DNA | a nucleotide sequence that directs the synthesis of a sequence complementary to it by the rules of *Watson crick base pairing*. A molecule that provides the structural mould to create similar molecules |
| the scientific method | A series of steps used for investigating observable events, acquiring new knowledge or correcting/integrating previous knowledge |
| transcriptome | The set of all *RNA* molecules, including *mRNA*, *rRNA*, *tRNA*, and other *non-coding RNA* produced in one or a population of cells |
| translocation | A misjoin of a scaffold/contig where the two pieces map to different chromosomes or plasmids |
| transport-RNA | Small RNA-molecules (ca. 73-94 nucleotides in length) used as the physical link between the nucleotide sequence of DNA/RNA and the amino acid sequence of proteins |
| tRNA | Abbrevation for *transport-RNA* |
| 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. |
| Watson crick base pairing | guanine-cytosine (G-C) and adenine-thymine (A-T) |

Appendix a  
(Plot/Figure)



Appendix A - Figure Structure of Galaxy instance

(Only altered folder/files are included)

A Separation line indicate that the instance is a folder

and an instance without the line is a file

Appendix B  
(Tables)

#### Recipe for reproduction of *Vibrio cholerae* data

|  |  |  |
| --- | --- | --- |
| Assembler | Read | Command |
| ABySS | HiSeq | abyss-pe k=51 l=1 n=5 s=100 name=asm lib='reads' \  reads=/Vibrio\_cholerae\_CO\_1032\_5\_HiSeq/trimmed/reads.trimmed.fastq \  aligner=bowtie |
|  | MiSeq | abyss-pe k=65 l=1 n=5 s=100 name=asm lib='reads' \  reads=/Vibrio\_cholerae\_CO\_0132\_5\_MiSeq/trimmed/reads.trimmed.fastq \  aligner=bowtie |
| CABOG | HiSeq | fastqToCA -insertsize 335 35 -libraryname reads -mates \  reads\_1.trimmed.fastq,reads\_2.trimmed.fastq > reads.frg  runCA -d . -p asm -s config reads.frg>&runCA.log |
|  | MiSeq | fastqToCA -insertsize 335 35 -libraryname reads -mates \  reads\_1.fastq,reads\_2.fastq > reads.frg  runCA -d . -p asm -s config reads.frg>&runCA.log |
| MIRA | HiSeq | runMira.sh which contains:  #!/bin/bash  numreads=800000  strainname="V.cholerae"  numlines=$((4\*${numreads}))  cat reads\_1.fastq | head -${numlines} | sed -e \  's/SRR[0-9.]\*/&\/1/'>${strainname}-${numreads}\_in.solexa.fastq  cat reads\_2.fastq | head -${numlines} | sed -e \  's/SRR[0-9.]\*/&\/2/'>${strainname}-${numreads}\_in.solexa.fastq  grep "@SRR" ${strainname}-${numreads}\_in.solexa.fastq | cut -f 1 -d ' ' | \ sed -e 's/@//' -e "s/$/ ${strainname}/" >> \  ${strainname}-${numreads}\_straindata\_in.txt  ln -s ${strainname}-${numreads}\_in.solexa.fastq mira\_in.solexa.fastq  ln -s ${strainname}-${numreads}\_straindata\_in.txt mira\_straindata\_in.txt  mira -fastq -job=denovo,genome,accurate,solexa -MI:sonfs=no:somrnl=0 \ SOLEXA\_SETTINGS -GE:tismin=167:tismax=502 -LR:file\_type=fastq \  -AS:mrpc=5>&log\_assembly.txt |
|  | MiSeq | runMira.sh which contains:  #!/bin/bash  numreads=800000  strainname="V.cholerae"  numlines=$((4\*${numreads}))  cat reads\_1.trimmed.fastq | head -${numlines} | sed -e \  's/SRR[0-9.]\*/&\/1/'>${strainname}-${numreads}\_in.solexa.fastq  cat reads\_2.trimmed.fastq | head -${numlines} | sed -e \  's/SRR[0-9.]\*/&\/2/'>${strainname}-${numreads}\_in.solexa.fastq  grep "@SRR" ${strainname}-${numreads}\_in.solexa.fastq | cut -f 1 -d ' ' | \  sed -e 's/@//' -e "s/$/ ${strainname}/" >> \  ${strainname}-${numreads}\_straindata\_in.txt  ln -s ${strainname}-${numreads}\_in.solexa.fastq mira\_in.solexa.fastq  ln -s ${strainname}-${numreads}\_straindata\_in.txt mira\_straindata\_in.txt  mira -fastq -job=denovo,genome,accurate,solexa -MI:sonfs=no:somrnl=0 \  SOLEXA\_SETTINGS -GE:tismin=167:tismax=502 -LR:file\_type=fastq \  -AS:mrpc=5>&log\_assembly.txt |
| MSRCA | HiSeq | runSRCA.pl config  assemble.sh  *where config file contains*  PATHS  JELLYFISH\_PATH=/full/path/to/MSR-CA-1.8.3/bin  SR\_PATH=/full/path/to/MSR-CA-1.8.3/bin  CA\_PATH=/full/path/to/CA-installation/bin  END  DATA  PE= p1 335 35 reads\_1.trimmed.fastq reads\_2.trimmed.fastq  END  PARAMETERS  GRAPH\_KMER\_SIZE=89 or 99  NUM\_THREADS=24  JF\_SIZE=2000000000  END |
|  | MiSeq | runSRCA.pl config  assemble.sh  *where config file contains*  PATHS  JELLYFISH\_PATH=/full/path/to/MSR-CA-1.8.3/bin  SR\_PATH=/full/path/to/MSR-CA-1.8.3/bin  CA\_PATH=/full/path/to/CA-installation/bin  END  DATA  PE= p1 335 35 reads\_1.trimmed.fastq reads\_2.trimmed.fastq  END  PARAMETERS  GRAPH\_KMER\_SIZE=89 or 99  NUM\_THREADS=24  JF\_SIZE=2000000000  END |
| SGA | HiSeq | module load samtools  module load gcc/4.8.0  ln -s/reads\_1.trimmed.fastq frag1  ln -s reads\_2.trimmed.fastq frag2  runSGA.sh  *which contains*  #!/bin/bash  K=65  CPU=8  MIN\_OVERLAP=45  ASSEMBLE\_OVERLAP=45  MIN\_PAIRS=5  sga preprocess --phred64 --pe-mode 1 -o reads.pp.fastq frag1 frag2  sga index --algorithm=ropebwt -t $CPU reads.pp.fastq  sga correct -k $K -t $CPU -o reads.ec.fastq reads.pp.fastq  sga index --algorithm=ropebwt -t $CPU reads.ec.fastq  sga filter -t $CPU reads.ec.fastq  sga overlap -m $MIN\_OVERLAP -t $CPU reads.ec.filter.pass.fa  sga assemble -o primary reads.ec.filter.pass.asqg.gz  ln -s primary-contigs.fa.ctg.fasta  bwa index ctg.fasta  bwa aln -t $CPU ctg.fasta frag1 > frag1.sai  bwa aln -t $CPU ctg.fasta frag2 > frag2.sai  bwa sampe ctg.fasta frag1.sai frag2.sai frag1 frag2 > frag.sam  samtools view –Sb frag.sam > libPE.bam  sga-bam2de.pl -n $MIN\_PAIRS --prefix libPE libPE.bam  sga-astat.py libPE.bam > libPE.astat  sga scaffold -m 200 -a libPE.astat -o scf --pe libPE.de ctg.fasta  sga scaffold2fasta -a primary-graph.asqg.gz -o scf.fasta scf |
|  | MiSeq | module load samtools  module load gcc/4.8.0  ln -s/reads\_1.trimmed.fastq frag1  ln -s reads\_2.trimmed.fastq frag2  runSGA.sh  *which contains:*  #!/bin/bash  K=65  CPU=8  MIN\_OVERLAP=85  ASSEMBLE\_OVERLAP=111  MIN\_PAIRS=5  sga preprocess --pe-mode 1 -o reads.pp.fastq frag1 frag2  sga index --algorithm=ropebwt -t $CPU reads.pp.fastq  sga correct -k $K -t $CPU -o reads.ec.fastq reads.pp.fastq  sga index --algorithm=ropebwt -t $CPU reads.ec.fastq  sga filter -t $CPU reads.ec.fastq  sga overlap -m $MIN\_OVERLAP -t $CPU reads.ec.filter.pass.fa  sga assemble -o primary reads.ec.filter.pass.asqg.gz  ln -s primary-contigs.fa.ctg.fasta  bwa index ctg.fasta  bwa aln -t $CPU ctg.fasta frag1 > frag1.sai  bwa aln -t $CPU ctg.fasta frag2 > frag2.sai  bwa sampe ctg.fasta frag1.sai frag2.sai frag1 frag2 > frag.sam  samtools view –Sb frag.sam > libPE.bam  sga-bam2de.pl -n $MIN\_PAIRS --prefix libPE libPE.bam  sga-astat.py libPE.bam > libPE.astat  sga scaffold -m 200 -a libPE.astat -o scf --pe libPE.de ctg.fasta  sga scaffold2fasta -a primary-graph.asqg.gz -o scf.fasta scf |
| SOAPdenovo2 + Gapcloser | HiSeq | SOAPdenovo-63mer all -K 51 -F -R -E -w -u -s config -o asm \  -p 8 >> SOAPdenovo.log  GapCloser -b config -a asm.scafSeq -o asm.new.scafSeq \  -t 8 >> SOAPdenovo.log  *Where config file contains*  [LIB]  avg\_ins=335  reverse\_seq=0  asm\_flags=3  rank=1  q1=reads\_1.trimmed.fastq  q2=reads\_2.trimmed.fastq |
|  | MiSeq | SOAPdenovo-63mer all -K 49 -F -R -E -w -u -s config -o asm \  -p 8 >> SOAPdenovo.log  GapCloser -b config -a asm.scafSeq -o asm.new.scafSeq \  -t 8 >> SOAPdenovo.log  *Where config file contains*  [LIB]  avg\_ins=335  reverse\_seq=0  asm\_flags=3  rank=1  q1=reads\_1.fastq  q2=reads\_2.fastq |
| SPAdes | HiSeq | spades.py -t 2 -k 33,55,65,75,85,99 \  --pe1-1 reads\_1.trimmed.fastq \  --pe1-2 reads\_2.trimmed.fastq \  -o output >spades.out 2>&1 |
|  | MiSeq | spades.py -t 2 -k 33,55,65,75,85,99 \  --pe1-1 reads\_1.trimmed.fastq \  --pe1-2 reads\_2.trimmed.fastq \  -o output >spades.out 2>&1 |
| Velvet | HiSeq | shuffleSequences\_fastq.pl reads\_1.trimmed.fastq reads\_2.trimmed.fastq \ reads.trimmed.fastq  velveth . 49 -fastq -shortPaired reads.trimmed.fastq  velvetg . -exp\_cov auto -ins\_length 335 -ins\_length\_sd 35 -scaffolding yes |
|  | MiSeq | shuffleSequences\_fastq.pl reads\_1.trimmed.fastq reads\_2.trimmed.fastq \ reads.trimmed.fastq  velveth . 97 -fastq -shortPaired reads.trimmed.fastq  velvetg . -exp\_cov auto -ins\_length 335 -ins\_length\_sd 35 -scaffolding yes |

Appendix B - Table 1 RECIPE FOR REPRODUCTION OF VIBRIO CHOLERAE DATA

#### Reproduction of gage-B result for *Vibrio cholerae*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembler | Version in  GAGE-B | Version in this thesis | Readtype for Miseq | Readtype for Hiseq |
| ABySS | 1.3.4 | 1.3.4 | Clean | Clean |
| CABOG | 7.0 | 7.0 8.1 | Raw Raw | Clean Clean |
| MIRA | 3.4.0 | 3.4.0 4.0.rc3 | Clean Clean | Raw Raw |
| MSRCA1 | 1.8.3 | 1.8.3  2.1.0 | Clean – K89 – gapclose error2 Clean – K99 – gapclose error2 Clean – K89 Clean – K99 | Raw – K89 Raw – K99 Raw – K89 Raw – K99 |
| SGA | 0.9.34 | 0.9.34 | Clean | Clean – phred64 error3 |
| SOAPdenovo2 + Gapcloser | 2.04  + 1.12 |  | Raw | Clean |
| SPAdes | 2.3.0 | 2.3.0 2.5.0 | Clean Clean | Clean Clean |
| Velvet | 1.2.08 | 1.2.08 1.2.10 | Clean Clean | Clean Clean |

Appendix B - Table 2 Raw reads are produced by the sequencer

clean (trimmed) reads are reads where adapter sequences are removed and q10 quality trimming using the ea-utils package is performed

1. The k-value in The gage-b recipe for msrca listed k=89 and k=99 for hiseq and nothing for miseq resulting in run with both k-values for both hiseq and miseq to determine the correct value
2. Example of gapclose error in Appendix B - table 3
3. example of phred64 error in Appendix B - table 3

#### Description of errors in Appendix B - Table 2

|  |  |
| --- | --- |
|  | Description |
| Gapclose error | Program: GapCloser  Version: 1.12  Parameters:  -a (scaffold file): asm.scafSeq  -b (config file): config  -o (output file): asm.new.scafSeq  -l (max read len): 100  -p (overlap para): 25  -t (thread num): 8  >>>>>>>>>>hash initializing<<<<<<<<<<  counting reads  in lib:  Error: can not open reads\_1.fastq  Program: GapCloser  Version: 1.12 |
| Phred64 error |  |

Appendix B - Table 3 Description of errors in APPENDIX B – Table 2

#### Problems discovered in GAGE-B recipe upon use

|  |  |  |
| --- | --- | --- |
| Type | Problem | Description |
| General |  | Bad choice of file-format (PDF). Should use plain text format to avoid character error while copy-pasting text |
| CABOG | Contig file specifying unitiger = bog | Typo: unitiger 🡪 unitigger |
| MIRA | **1)** ”NA” | **1)** “Bad quotation mark” used. |
|  | **2)** "cat reads2.fastq | head -${numlines} | sed -e 's/SRR[0-9.]\*/&\/1/' >${strainname}-${numreads} in.solexa.fastsq" | **2)** Typo1: It should be 's/SRR[0-9.]\*/&\/2/'  Typo2: fastsq 🡪 fastq |
|  | **3)** "cat reads1.fastq | head -${numlines} | sed -e 's/SRR[0-9.]\*/&\/1/' >${strainname}-${numreads} in.solexa.fastsq" | **3)** Typo1: Missing \_ before in.solexa.fastsq  Typo2: fastsq 🡪 fastq |
|  | **4)** "with srrname and numreads containing the corrects values for each run" | **4)** Where to find the srrname? It looks like it isn’t used at all, except for the initialization |
| SGA | **1)** Phred-error | **1)** Running SGA on Vibrio cholera Hiseq data according to the recipe result in phred64-error |
|  | **2)** 'GLIBCXX\_2.4.15 not found' during a run | **2)** Need to load gcc module for this run. It would be nice to know the dependencies prior to a run |
| MSRCA | **1)** k-value is 89 **and** 99 for Vibrio cholerae HiSeq data | **1)** Typo: Which K-value is for Miseq data? |
|  | **2)** Config file contains “NUM\_THREADS=t” | **2)** Cannot find what t is. |

Appendix B - Table 4 Some Problems discovered in the GAGE-B recipe upon use

#### Assembly statistics for *Vibrio cholerae*

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 200 bp)" and "Total length (>= 200 bp)" include all contigs). Best result for each metric is written in bold. The first column (GAGE-B supp) refers to data gotten from GAGE-Bs supplementary material apr4. The values for N50/NA50 from the supplementary material is written in kbp and is extracted to total number of bp, making the statistics accurate to the nearest hundredth. The second column marked as “GAGE-B” refer to assemblies downloaded from <http://ccb.jhu.edu/gage_b/genomeAssemblies/index.html> with various assemblers as described in *Appendix B – Table 2 – Reproduction of GAGE-B results for Vibrio cholerae*. There’s a difference between two GAGE-B values (supplementary vs downloaded) if the adjacent cells are coloured red. A cell is marked with the value N/A if there is no data available for that given metric. For the duplication ratio in GAGE-B supplementary material, since the number seems to be rounded up to 1 decimal, it has been excluded from the comparison of best result unless all the other values exceed 1.0.

#### Assembly statistics for *Vibrio cholerae* on CABOG with hiseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Cabog 7.0 | Cabog 8.1 |
| # contigs (>= 200 bp) | **127** | **127** | 144 | 519 |
| N50 | 57 900 | **61 249** | 57 089 | 10 437 |
| NA50 | 48 800 | **57 813** | 57 089 | 10 437 |
| # misassemblies | 33 | 20 | 10 | **8** |
| # local misassemblies | 12 | 11 | **7** | **7** |
| # fully unaligned contigs | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **96.6** | 95.623 | 95.361 | 91.790 |
| Duplication ratio | 1.0 | 1.007 | 1.006 | **1.002** |
| # genes | N/A | **3 374** | 3 346 | 2 970 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. cholerae on CABOG (Hiseq – contig)

#### Assembly statistics for *Vibrio cholerae* on CABOG with Hiseq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B  supp | GAGE-B | Cabog 7.0 | Cabog 8.1 |
| # scaffold (>= 500 bp) | 108 | 108 | **94** | 183 |
| N50 | 67 000 | 67 078 | **134 075** | 38 698 |
| NA50 | 53 200 | 63 201 | **134 075** | 38 209 |
| # misassemblies | 34 | 21 | 11 | **10** |
| # local misassemblies | 24 | 23 | **13** | 38 |
| # fully unaligned contigs | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **96.6** | 95.629 | 95.321 | 92.029 |
| Duplication ratio | 1.0 | **1.012** | 1.023 | 1.044 |
| # genes | N/A | 3 380 | 3 383 | 3 185 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON CABOG (HISEQ – Scaffold)

#### Assembly statistics for *Vibrio cholerae* on CABOG with Miseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B (cabog 7.0) | Cabog 7.0 | Cabog 8.1 |
| # contigs (>= 200 bp) | 241 | 241 | **188** | 286 |
| N50 | 32 800 | **33 710** | 1 707 | 24 969 |
| NA50 | 32 500 | **33 710** | 1 624 | 24 969 |
| # misassemblies | 22 | 17 | **7** | 8 |
| # local misassemblies | 7 | 7 | **1** | 6 |
| # fully unaligned contigs | 1 | 1 | **0** | **0** |
| Genome fraction (%) | **97.8** | 96.968 | 7.639 | 93.765 |
| Duplication ratio | 1.0 | 1.016 | 1.011 | **1.009** |
| # genes | N/A | **3 401** | 123 | 3 286 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON CABOG (MISEQ – CONTIG)

#### Assembly statistics for *Vibrio cholerae* on CABOG with Miseq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Cabog 7.0 | Cabog 8.1 |
| # scaffold (>= 500 bp) | 241 | 241 | **188** | 285 |
| N50 | 32 800 | **33 710** | 1 707 | 24 969 |
| NA50 | 32 500 | **33 710** | 1 624 | 24 969 |
| # misassemblies | 22 | 17 | **7** | 9 |
| # local misassemblies | 7 | 7 | **1** | 6 |
| # fully unaligned contigs | 1 | 1 | **0** | **0** |
| Genome fraction (%) | **97.8** | 96.968 | 7.639 | 93.765 |
| Duplication ratio | 1.0 | 1.016 | 1.011 | **1.009** |
| # genes | N/A | **3 401** | 123 | 3 286 |

#### Assembly statistics for *Vibrio cholerae* on MIRA with hiseq data (contig)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MIRA 3.4.0 |
| # contigs (>= 200 bp) | 728 | **733** | 1524 |
| N50 | **92 000** | **92 000** | 4 840 |
| NA50 | 87 100 | **89 505** | 4 812 |
| # misassemblies | 89 | **24** | 45 |
| # local misassemblies | 15 | **9** | 10 |
| # fully unaligned contigs | 10 | **0** | **0** |
| Genome fraction (%) | **99.7** | 97.925 | 94.917 |
| Duplication ratio | 1.0 | 1.016 | **1.009** |
| # genes | N/A | **3 516** | 2 627 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MIRA (HISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on MIRA with Miseq data (contig)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MIRA 3.4.0 |
| # scaffold (>= 500 bp) | 430 | 431 | **224** |
| N50 | **112 900** | **112 926** | 108 646 |
| NA50 | **108 700** | **108 689** | 108 646 |
| # misassemblies | 148 | 49 | **23** |
| # local misassemblies | 17 | 7 | **4** |
| # fully unaligned contigs | 20 | 5 | **1** |
| Genome fraction (%) | **99.6** | 98.311 | 98.078 |
| Duplication ratio | 1.0 | 1.016 | **1.012** |
| # genes | N/A | **3 559** | 3 534 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MIRA (MISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on MaSuRCA with hiseq data (contig)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MaSuRCA 1.8.3 k89 | MaSuRCA  1.8.3 k99 | MaSuRCA  2.1.0 k89 | MaSuRCA  2.1.0 k99 |
| # contigs (>= 200 bp) | **105** | **105** | 137 | 330 | 179 | 119 |
| N50 | **241 600** | **241 604** | 108 823 | 35 282 | 93 473 | 2 355 |
| NA50 | **236 400** | **236 373** | 90 882 | 35 282 | 81 529 | 2 208 |
| # misassemblies | 12 | 8 | 8 | 11 | 8 | **2** |
| # local misassemblies | 5 | 5 | 4 | 5 | 6 | **0** |
| # fully unaligned contigs | **0** | **0** | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **99.4** | 98.147 | 98.121 | 95.714 | 98.498 | 4.709 |
| Duplication ratio | 1.0 | 1.013 | 1.008 | **1.003** | 1.009 | 1.004 |
| # genes | N/A | **3 573** | 3 552 | 3 289 | 3 541 | 94 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MaSuRCA (HISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on MaSuRCA with hiseq data (scaffold)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MaSuRCA 1.8.3 k89 | MaSuRCA  1.8.3 k99 | MaSuRCA  2.1.0 k89 | MaSuRCA  2.1.0 k99 |
| # scaffold (>= 500 bp) | **88** | **88** | **88** | 245 | 118 | 167 |
| N50 | 246 500 | 246 505 | **246 836** | 46 624 | 133 199 | 65 767 |
| NA50 | 236 400 | 236 373 | **236 430** | 46 604 | 132 265 | 65 252 |
| # misassemblies | 11 | **9** | 10 | 12 | 13 | 11 |
| # local misassemblies | 8 | 7 | 15 | 31 | 20 | **6** |
| # fully unaligned contigs | **0** | **0** | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **99.3** | 98.147 | 98.152 | 95.804 | 98.498 | 96.353 |
| Duplication ratio | 1.0 | 1.013 | 1.013 | **1.009** | 1.010 | 1.027 |
| # genes | N/A | **3 573** | 3 568 | 3 314 | 3 557 | 3 480 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MaSuRCA (HISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on MaSuRCA with Miseq data (contig)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MaSuRCA 1.8.3 k89 | MaSuRCA  1.8.3 k99 | MaSuRCA  2.1.0 k89 | MaSuRCA  2.1.0 k99 |
| # contig (>= 200 bp) | **173** | **173** | 174 | **173** | 182 | 182 |
| N50 | **76 100** | **76 131** | 61 317 | **76 131** | 62 007 | 65 767 |
| NA50 | **71 600** | **76 131** | 60 468 | **76 131** | 61 330 | 65 252 |
| # misassemblies | 23 | **19** | 20 | **19** | 11 | 9 |
| # local misassemblies | 5 | **3** | 5 | **3** | 5 | 6 |
| # fully unaligned contigs | **0** | **0** | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **98.3** | 97.670 | 96.864 | 97.670 | 96.147 | 96.349 |
| Duplication ratio | 1.0 | **1.023** | 1.027 | **1.023** | 1.029 | 1.027 |
| # genes | N/A | **3 534** | 3 500 | **3 534** | 3 474 | 3 480 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MaSuRCA (MISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on MaSuRCA with Miseq data (scaffold)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | MaSuRCA 1.8.3 k89 | MaSuRCA  1.8.3 k99 | MaSuRCA  2.1.0 k89 | MaSuRCA  2.1.0 k99 |
| # scaffold (>= 500 bp) | **163** | **163** | 167 | **163** | 171 | 167 |
| N50 | **76 100** | **76 131** | 61 317 | **76 131** | 62 007 | 65 767 |
| NA50 | 71 600 | **76 131** | 60 468 | **76 131** | 61 330 | 65 252 |
| # misassemblies | 23 | 19 | 20 | 19 | 13 | **11** |
| # local misassemblies | 5 | **3** | 5 | **3** | 5 | 6 |
| # fully unaligned contigs | **0** | **0** | **0** | **0** | **0** | **0** |
| Genome fraction (%) | **98.3** | 97.670 | 96.864 | 97.670 | 96.147 | 96.353 |
| Duplication ratio | 1.0 | **1.023** | 1.027 | **1.023** | 1.029 | 1.027 |
| # genes | N/A | **3 534** | 3 500 | **3 534** | 3 474 | 3 480 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON MaSuRCA (MISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on SOAPdenovo with hiseq data (contig)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | SOAPdenovo2 |
| # contigs (>= 200 bp) | **139** | **139** | 462 |
| N50 | 125 900 | **135 118** | 21 660 |
| NA50 | 106 500 | **112 904** | 21 657 |
| # misassemblies | 26 | 15 | **2** |
| # local misassemblies | 50 | 50 | **0** |
| # fully unaligned contigs | 5 | **1** | **1** |
| Genome fraction (%) | **99.5** | 97.295 | 96.488 |
| Duplication ratio | 1.0 | 1.009 | **1.002** |
| # genes | N/A | **3 479** | 3 274 |

Appendix B - Table ASSEMBLY STATISTICS FOR   
V. CHOLERAE ON SOAPdenovo (HISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on SOAPdenovo with hiseq data (scaffold)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | SOAPdenovo2 |
| # scaffold (>= 500 bp) | **75** | **75** | 77 |
| N50 | 181 100 | 200 529 | **200 760** |
| NA50 | 168 100 | 181 115 | **181 222** |
| # misassemblies | 26 | 15 | **6** |
| # local misassemblies | **76** | **76** | 81 |
| # fully unaligned contigs | **1** | **1** | **1** |
| Genome fraction (%) | **99.0** | 97.305 | 97.517 |
| Duplication ratio | 1.0 | **1.009** | 1.011 |
| # genes | N/A | 3 480 | **3 485** |

Appendix B - Table ASSEMBLY STATISTICS FOR   
V. CHOLERAE ON SOAPdenovo (HISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on SOAPdenovo with Miseq data (contig)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B | GAGE-B | SOAPdenovo2 |
| # contigs (>= 200 bp) | **244** | **244** | 439 |
| N50 | **71 400** | **71 357** | 29 551 |
| NA50 | 65 500 | **71 357** | 29 551 |
| # misassemblies | 21 | 12 | **2** |
| # local misassemblies | 48 | 44 | **0** |
| # fully unaligned contigs | 4 | **2** | **2** |
| Genome fraction (%) | **99.3** | 96.940 | 96.230 |
| Duplication ratio | 1.0 | 1.003 | **1.001** |
| # genes | N/A | **3 442** | 3 336 |

Appendix B - Table ASSEMBLY STATISTICS FOR   
V. CHOLERAE ON SOAPdenovo (MISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on SOAPdenovo with Miseq data (scaffold)

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | SOAPdenovo2 |
| # scaffold (>= 500 bp) | **165** | **165** | 166 |
| N50 | 91 900 | 91 942 | **92 055** |
| NA50 | 89 900 | 91 942 | **92 054** |
| # misassemblies | 24 | 14 | **6** |
| # local misassemblies | 80 | **77** | 111 |
| # fully unaligned contigs | **1** | 2 | 2 |
| Genome fraction (%) | **98.7** | 97.050 | 97.068 |
| Duplication ratio | 1.0 | **1.005** | **1.005** |
| # genes | N/A | **3 443** | 3 432 |

Appendix B - Table ASSEMBLY STATISTICS FOR   
V. CHOLERAE ON SOAPdenovo (MISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on SPAdes with Hiseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Spades 2.3.0 | Spades 2.5.0 |
| # contigs (>= 200 bp) | 205 | 205 | 158 | **134** |
| N50 | 77 100 | 83 518 | 137 662 | **225 872** |
| NA50 | 77 100 | 83 518 | 137 656 | **197 838** |
| # misassemblies | 7 | **4** | **4** | 8 |
| # local misassemblies | 4 | **2** | 9 | 3 |
| # fully unaligned contigs | 8 | **1** | 3 | 3 |
| Genome fraction (%) | **99.6** | 97.439 | 98.611 | 97.468 |
| Duplication ratio | 1.0 | 1.007 | 1.007 | **1.006** |
| # genes | N/A | 3 483 | **3 571** | 3 519 |

Appendix B - Table Assembly statistics for V. cholerae on SPAdes (Hiseq contig)

#### Assembly statistics for *Vibrio cholerae* on SPAdes with Hiseq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Spades 2.3.0 | Spades 2.5.0 |
| # scaffold (>= 500 bp) | 106 | 106 | 109 | **93** |
| N50 | 98 300 | 98 274 | 225 904 | **343 954** |
| NA50 | 94 800 | 95 858 | 214 751 | **246 236** |
| # misassemblies | 27 | 21 | **7** | 8 |
| # local misassemblies | 19 | 17 | 11 | **6** |
| # fully unaligned contigs | **1** | **1** | 3 | 3 |
| Genome fraction (%) | **99.6** | 98.209 | 98.753 | 97.478 |
| Duplication ratio | 1.0 | 1.021 | 1.062 | **1.011** |
| # genes | N/A | 3 544 | **3 586** | 3 524 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON SPAdes (HISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on SPAdes with Miseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Spades 2.3.0 | Spades 2.5.0 |
| # contigs (>= 200 bp) | 1475 | 1475 | 1480 | **786** |
| N50 | **262 200** | **262 160** | **262 160** | 246 502 |
| NA50 | 246 600 | **262 160** | **262 160** | 198 500 |
| # misassemblies | 7 | 5 | 5 | **4** |
| # local misassemblies | 6 | 4 | 4 | **1** |
| # fully unaligned contigs | 1336 | **57** | **57** | 81 |
| Genome fraction (%) | **99.6** | 98.643 | 98.752 | 97.350 |
| Duplication ratio | 1.0 | 1.004 | 1.004 | **1.001** |
| # genes | N/A | **3 598** | 3 597 | 3 505 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON SPAdes (MISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on SPAdes with Miseq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Spades 2.3.0 | Spades 2.5.0 |
| # scaffold (>= 500 bp) | **145** | **145** | 147 | 184 |
| N50 | **262 200** | **262 160** | **262 160** | 258 677 |
| NA50 | 246 600 | **262 160** | **262 160** | 215 247 |
| # misassemblies | 7 | 5 | 5 | **4** |
| # local misassemblies | 6 | 4 | 4 | **3** |
| # fully unaligned contigs | **57** | **57** | **57** | 81 |
| Genome fraction (%) | **99.6** | 98.648 | 98.752 | 97.356 |
| Duplication ratio | 1.0 | 1.004 | 1.004 | **1.001** |
| # genes | N/A | **3 599** | 3 598 | 3 508 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON SPAdes (MISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on Velvet with Hiseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Velvet 1.2.08 | Velvet 1.2.10 |
| # contigs (>= 200 bp) | 261 | 261 | **246** | **246** |
| N50 | 40 100 | 40 877 | **46 346** | **46 346** |
| NA50 | 39 500 | 40 877 | **42 805** | **42 805** |
| # misassemblies | 9 | **4** | 9 | 9 |
| # local misassemblies | 9 | **8** | 9 | 9 |
| # fully unaligned contigs | **1** | **1** | **1** | **1** |
| Genome fraction (%) | **99.4** | 97.038 | 96.340 | 96.340 |
| Duplication ratio | 1.0 | **1.011** | 1.012 | 1.012 |
| # genes | N/A | 3 386 | **3 392** | **3 392** |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON Velvet (HISEQ CONTIG)

#### Assembly statistics for *Vibrio cholerae* on Velvet with Hiseq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Velvet 1.2.08 | Velvet 1.2.10 |
| # scaffold (>= 500 bp) | **85** | **85** | 104 | 104 |
| N50 | **172 500** | **172 545** | 163 386 | 163 386 |
| NA50 | **171 500** | **171 505** | 163 055 | 163 055 |
| # misassemblies | 13 | **10** | **10** | **10** |
| # local misassemblies | 132 | 129 | **123** | **123** |
| # fully unaligned contigs | **1** | **1** | **1** | **1** |
| Genome fraction (%) | **98.9** | 97.140 | 96.251 | 96.251 |
| Duplication ratio | 1.0 | **1.016** | **1.016** | **1.016** |
| # genes | N/A | **3 400** | 3 379 | 3 379 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON VELVET (HISEQ scaffold)

#### Assembly statistics for *Vibrio cholerae* on Velvet with Miseq data (contig)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Velvet 1.2.08 | Velvet 1.2.10 |
| # contigs (>= 200 bp) | 201 | 201 | **179** | **179** |
| N50 | 92 000 | 92 036 | **105 176** | **105 176** |
| NA50 | 67 100 | 67 096 | **105 176** | **105 176** |
| # misassemblies | 12 | 14 | **5** | **5** |
| # local misassemblies | 7 | **2** | 3 | 3 |
| # fully unaligned contigs | **1** | **1** | **1** | **1** |
| Genome fraction (%) | **99.5** | 97.563 | 96.234 | 96.234 |
| Duplication ratio | 1.0 | **1.007** | **1.007** | **1.007** |
| # genes | N/A | **3 491** | 3 460 | 3 460 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON VELVET (Miseq contig)

#### Assembly statistics for *Vibrio cholerae* on Velvet with MiSeq data (scaffold)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assembly | GAGE-B supp | GAGE-B | Velvet 1.2.08 | Velvet 1.2.10 |
| # scaffold (>= 500 bp) | 138 | 138 | **133** | **133** |
| N50 | **110 000** | **109 996** | 105 176 | 105 176 |
| NA50 | 92 000 | 75 901 | **105 176** | **105 176** |
| # misassemblies | 17 | 22 | **6** | **6** |
| # local misassemblies | 23 | 13 | **6** | **6** |
| # fully unaligned contigs | **1** | **1** | **1** | **1** |
| Genome fraction (%) | **99.2** | 97.598 | 96.242 | 96.242 |
| Duplication ratio | 1.0 | **1.007** | **1.007** | **1.007** |
| # genes | N/A | **3 492** | 3 459 | 3 459 |

Appendix B - Table ASSEMBLY STATISTICS FOR V. CHOLERAE ON VELVET (MISEQ scaffold)

References

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

*Aggressive assembly of pyrosequencing reads with mates.*

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

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

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

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

3. J. T. Simpson, 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. A. Gurevich, V. Saveliev, N. Vyahhi, et al.,

*Quast: Quality assessment tool for genome assemblies.*

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

5. S. L. Salzberg, 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. K. Bradnam, J. Fass, A. Alexandrov, et al.,

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

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

7. D. Earl, 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. Genome assembly

Available from: <http://www.k.u-tokyo.ac.jp/pros-e/person/shinichi_morishita/shinichi_morishita.htm>

9. W.-K. Sung,

*Algorithms in bioinformatics: A practical introduction*. 2009: CRC Press (Taylor & Francis Group). 24-25.

10. M. L. Metzker,

*Sequencing technologies - the next generation.*

Nature Reviews Genetics, 2010. **11**(1): p. 31-46.

11. M. C. Schatz, A. L. Delcher, and S. L. Salzberg,

*Assembly of large genomes using second-generation sequencing.*

Genome Res, 2010. **20**(9): p. 1165-73.

12. M. Pop,

*Genome assembly reborn: Recent computational challenges.*

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

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

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

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

14. J. Shendure and H. Ji,

*Next-generation DNA sequencing.*

Nature Biotechnology, 2008. **26**(10): p. 1135-1145.

15. S. Schlebusch and N. Illing,

*Next generation shotgun sequencing and the challenges of de novo genome assembly.*

South African Journal of Science, 2012. **108**(11-12): p. 8.

16. S. L. Salzberg and J. A. Yorke,

*Beware of mis-assembled genomes.*

Bioinformatics, 2005. **21**(24): p. 4320–21.

17. Next generation sequencing: An overview Access date: 26.05.2014

Available from: [http://www.corning.com/uploadedFiles/Lifesciences/PDFs/Axygen\_PDFs/NGS%20Overview.pdf^^](http://www.corning.com/uploadedFiles/Lifesciences/PDFs/Axygen_PDFs/NGS%20Overview.pdf%5e%5e)

18. R. Barthelson, A. J. McFarlin, S. D. Rounsley, et al.,

*Plantagora: Modeling whole genome sequencing and assembly of plant genomes.*

PLoS ONE, 2011. **6**(12): p. Table S1.

19. Reproducibility Access date: 08.04.14

Available from: [http://todayinsci.com/K/Kohn\_Alexander/KohnAlexander-Quotations.htm^^](http://todayinsci.com/K/Kohn_Alexander/KohnAlexander-Quotations.htm%5e%5e)

20. R. D. Peng, F. Dominici, and S. L. Zeger,

*Reproducible epidemiologic research.*

American Journal of Epidemiology, 2006. **163**(9): p. 783-789.

21. G. K. Sandve, A. Nekrutenko, J. Taylor, et al.,

*Ten simple rules for reproducible computational research.*

PLOS Computational Biology, 2013. **9**(10): p. e1003285.

22. E. S. Raymond,

*The cathedral and the bazaar*. 1998: O'Reilly Media.

23. R. Prieto-Diaz,

*Status report: Software reusability.*

Software, IEEE, 1993. **10**(3): p. 61-66.

24. Galaxy mailing lists Access date: Available from: https://wiki.galaxyproject.org/MailingLists^^

25. A. M. Bolger, M. Lohse, and B. Usadel,

*Trimmomatic: A flexible trimmer for illumina sequence data.*

Bioinformatics, 2014.

26. Abyss webpage Access date: Available from: [http://www.bcgsc.ca/platform/bioinfo/software/abyss^^](http://www.bcgsc.ca/platform/bioinfo/software/abyss%5e%5e)

27. E. W. Myers, G. G. Sutton, A. L. Delcher, et al.,

*A whole-genome assembly of drosophila.*

Science, 2000. **287**(5461): p. 2196-2204.

28. B. Chevreux, T. Wetter, and S. Suhai. *Genome sequence assembly using trace signals and additional sequence information*. in *Computer Science and Biology: Proceedings of the German Conference on Bioinformatics (GCB)*. 1999.

29. A. V. Zimin, G. Marcais, D. Puiu, et al.,

*The masurca genome assembler.*

Bioinformatics, 2013. **29**(21): p. 2669-77.

30. J. T. Simpson and R. Durbin,

*Efficient de novo assembly of large genomes using compressed data structures.*

Genome Research, 2012. **22**(3): p. 549-56.

31. R. Luo, B. Liu, Y. Xie, et al.,

*Soapdenovo2: An empirically improved memory-efficient short-read de novo assembler.*

Gigascience, 2012. **1**(1): p. 18.

32. A. Bankevich, S. Nurk, D. Antipov, et al.,

*Spades: A new genome assembly algorithm and its applications to single-cell sequencing.*

Journal of computational biology : a journal of computational molecular cell biology, 2012. **19**(5): p. 455-77.

33. What is python Access date: 28.05.2014

Available from: https://docs.python.org/2/faq/general.html#what-is-python^^

34. Quast Access date: 28.05.2014

Available from: [http://quast.bioinf.spbau.ru/^^](http://quast.bioinf.spbau.ru/%5e%5e)

35. Google chart Access date: 28.05.2014

Available from: [http://en.wikipedia.org/wiki/Google\_Chart\_API^^](http://en.wikipedia.org/wiki/Google_Chart_API%5e%5e)

36. Sqlite3 Access date: 28.05.2014

Available from: https://docs.python.org/2/library/sqlite3.html^^

37. Os module Access date: 28.05.2014

Available from: [http://www.pythonforbeginners.com/os/pythons-os-module^^](http://www.pythonforbeginners.com/os/pythons-os-module%5e%5e)

38. Json in python Access date: 28.05.2014

Available from: https://docs.python.org/2/library/json.html^^

39. Time module Access date: 28.05.2014

Available from: https://docs.python.org/2/library/time.html^^

40. Zipfile module Access date: 28.05.2014

Available from: https://docs.python.org/2/library/zipfile.html^^

41. Pypdf Access date: 28.05.2014

Available from: https://pypi.python.org/pypi/pyPdf^^

42. Reportlab toolkit Access date: 28.05.2014

Available from: https://pypi.python.org/pypi/reportlab/3.1.8^^

43. Reportlab user guide Access date: 28.05.2014

Available from: [http://www.reportlab.com/docs/reportlab-userguide.pdf^^](http://www.reportlab.com/docs/reportlab-userguide.pdf%5e%5e)

44. Collections module Access date: 28.05.2014

Available from: https://docs.python.org/2/library/collections.html^^

1. https://wiki.galaxyproject.org/ [↑](#footnote-ref-1)