UiO : Department of Informatics
University of Oslo

Reproducibility and reusability of genome assembly evaluation

Master's thesis

Sabba Ifzal

Spring, 2014

REPRODUCIBILITY AND REUSABILITY OF GENOME ASSEMBLY EVALUATION

SABBA IFZAL

JULY 15, 2014

ACKNOWLEDGEMENTS

This thesis represent not only pages upon pages with results, discussions and conclusions, but several semesters' worth of private lectures, guidance, and support from a lot of people. My experience with the Biomedical Informatics research group at the Department of Informatics have been nothing but highly educational and a lot of fun. I have been given the opportunity to experience the amazing field of genome sequencing and assembly and to learn from the very best.

A lot of people need to be acknowledged and I would like start by thanking my main supervisor, Torbjørn Rognes, for his patient guidance and encouragement during my time as his student. He even proof read my thesis during his vacation and I cannot thank him enough for that. I would also like to thank my first co-supervisor, Alexander Johan Nederbragt, for giving me private lectures in biology about everything from the very basics, such as DNA and RNA, to the more complex understanding of genome sequencing and assembly. He has been like my own encyclopedia and I don't know how I would have been able to complete this thesis without his passion and knowledge in biology. All credits to my second co-supervisor, Geir Kjetil Sandve, for the help I got during the implementation of the Galaxy tools. I always got an answer to my question whether he answered it himself or asked someone else to help me. The Galaxy tools would not have been implemented the way they are if not for him. At last, my third co-supervisor, Ksenia Khelik, my biggest thanks to her for regularly checking up on me, mentally preparing me for an upcoming oral presentation, and helping me to limit the boundaries of this thesis.

Completing this work would have been a lot more difficult were it not for the help from a lot of people at the Department of Informatics and CEES, especially Sveinung Gundersen, Kai Trengereid, Jon K. Lærdahl, Morten Johansen and Ole Kristian Tørresen.

I also want to thank my family for being there for me in my ups and downs and always believing in me. More than once have the support from them helped me through a difficult time and I'm forever indebted to them for that.

Finally, I would like to express my gratitude to my husband for his support, encouragement, and for never complaining about the pile of laundry or dirty dishes begging to be washed. He also always bought midnight snacks to satisfy my cravings during my nightly writing sessions, and proof read my thesis from time to time. I cannot thank him enough for everything he's done for me during my time as a student.

Sabba Ifzal
University of Oslo
July, 2014

ABSTRACT

DNA sequencing technology such as Next Generation Sequencing (NGS) is developing and revolutionizing the field of sequencing, allowing scientists to determine the sequence of nucleotides with an extreme speed. The task of puzzling together small pieces of a sequence from a new genome into larger continuous parts is difficult, and the bioinformatics field lacks enough information about which method would perform best under certain conditions.

The aims for this thesis is to create a system or tool which enable its users to assess assemblies based on existing technology, such as QUAST [1], but with results that can be visualized with more custom, user-defined features, such as bar charts and scatterplots. This thesis will also tentatively reproduce results from the GAGE-B paper [2], as well as reuse the same data with newer versions of the assemblers to assess any development experienced on bacterial genomes.

It was more difficult than anticipated to reproduce results, mostly because of unsatisfying descriptions in the GAGE-B paper, but the results showed that despite the numerical differences observed, the conclusion from the GAGE-B paper was not significantly changed. Experiments also showed that the new Galaxy tools developed for assembly evaluation can be helpful for the scientific community to make easily reproducible data and for comparison of assemblies in the future.

PREFACE

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

This project started out with a reproduction of assemblies over all species used in the GAGE-B paper, but was later, due to lack of time and satisfying descriptions in the paper, reduced to mainly focus on assemblies performed on *Vibrio cholerae*.

The Galaxy tools for assembly evaluation can be used from http://insilico.hpc.uio.no:24688 while the code is accessible from the following Github repository: https://github.com/subway/Galaxy-Distribution.

Supplementary materials used in this thesis are:

- Recipe (used for each assembler)
- Supplementary_Tables (with more information about the assemblies)
    - Referred to as Table S in this thesis
- Supplementary_Figures_A (figures related to inconsistent GAGE-B results)
    - Referred to as Figure S-A in this thesis
- Supplementary_Figures_B (figures related to the reproduced results)
    - Referred to as Figure S-B in this thesis

All the supplementary materials are available on Github and can be accessed from https://github.com/subway/masterthesis/tree/master/Supplementary%20Material.

Note that there is a glossary towards the end of this thesis containing information about certain words and phrases written in **bold** throughout this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 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 [3], Velvet [4] and ABySS [5] 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 [1], 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[6], GAGE-B[2] and Assemblathon 1 and 2 [7, 8], 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.

One of the desired features that current benchmarks are weak on is the ability to visualize results in charts of various types. Another desire might be to have a system or tool that is technically advanced, but user-friendly so that less experienced computer-users can easily adapt to the use of the system or tool. 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 or tool which can give scientist the opportunity to reuse data to compare old against new versions with minimal effort is highly wished for. The same goes for when scientist want to reproduce results from either earlier computations or perhaps published articles.

The desired outcome for a new system or tool 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.

1.2 PROBLEM STATEMENT / AIMS

The aims for this thesis is to create a system or tool which enable its users to assess assemblies based on existing technology, such as QUAST, but with results that can be visualized with more custom, user-defined features, such as bar charts and scatterplots. These features can be useful for determining the best performing assembler based on what the user see rather than a lot of numbers to be manually compared upon. The system or tool will hopefully make it easier to reuse datasets, assemblies and compared results, as well as making comparison of reproduced results a piece of cake. It will also make it possible to have the same approach for the (same) data no matter how long it's been since the last approach.

This thesis will also aim to reproduce GAGE-B results, as well as reuse the same data with newer versions of the assembler to assess any development experienced on bacterial genomes. Results

from both reproducing and reusing data will be subject to comparison and assessment using the new system/tool developed as part of this thesis.

1.3 PROBLEM SOLUTION

The solution to the increasing need of new system with increased visualization features that maintained the reusability, and simplified the assessment of reproduced results 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.

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: *insilico.hpc.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 tools on their own server, then all they need to do is copy the tool folder from https://github.com/subway/Galaxy-Distribution to their Galaxy instance and add proper links to the tools in the tool_conf.xml.

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.

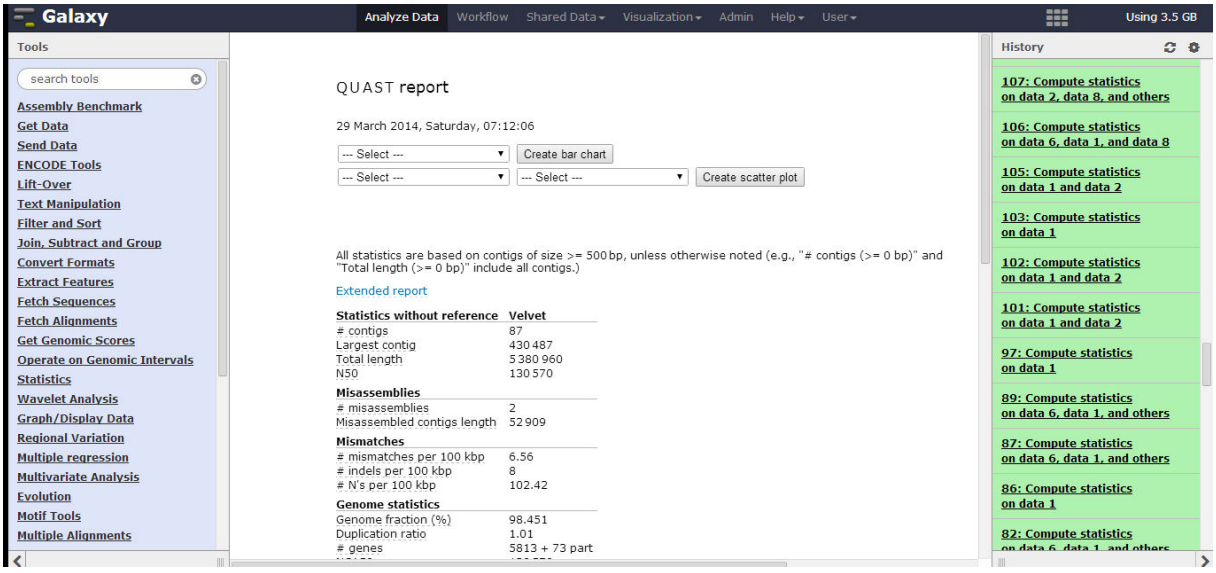


FIGURE 1-1 SCREENSHOT OF THE GALAXY TOOL WITH AN EXAMPLE OF THE HISTORY PANEL
Source: http://insilico.hpc.uio.no:24688/

CHAPTER 2 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.

2.1 GENOME SEQUENCING AND ASSEMBLY

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

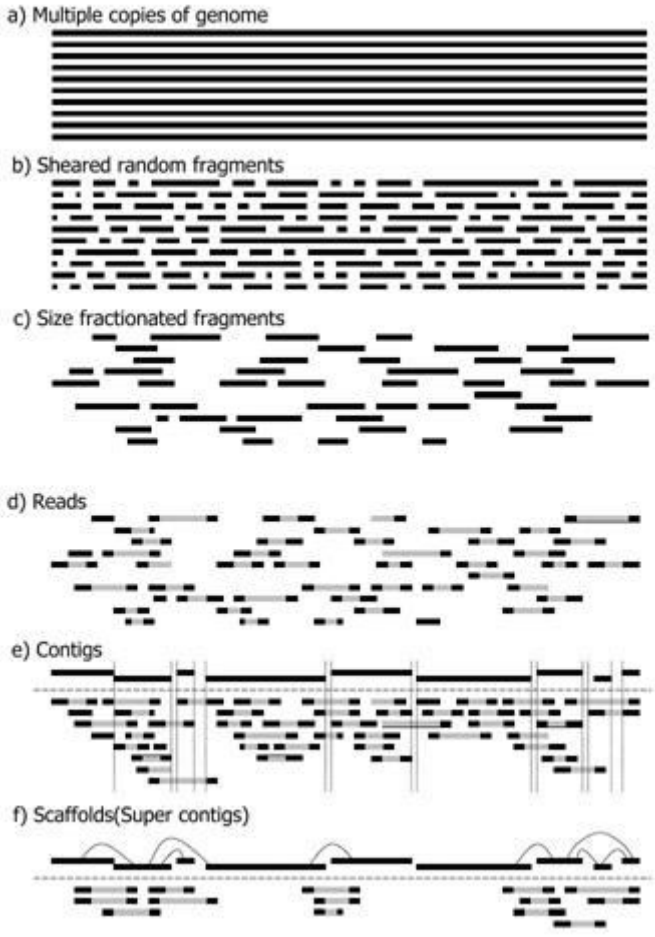


FIGURE 2-1 BASIC STEPS OF GENOME SEQUENCING AND ASSEMBLY
Source: Morishita [9]

2.1.1 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

Many sequencing technologies such as PacBio, Ion Torrent and Illumina are used today. Even though a short introduction to the most well-known might be desired for a newcomer in the field of bioinformatics only the sequencing technology used in this thesis will be briefly described. Information used in this section is gathered from the book *Algorithms in Bioinformatics* [10].

The technology was acquired and commercialized by Illumina in 2006 and 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 2-2)

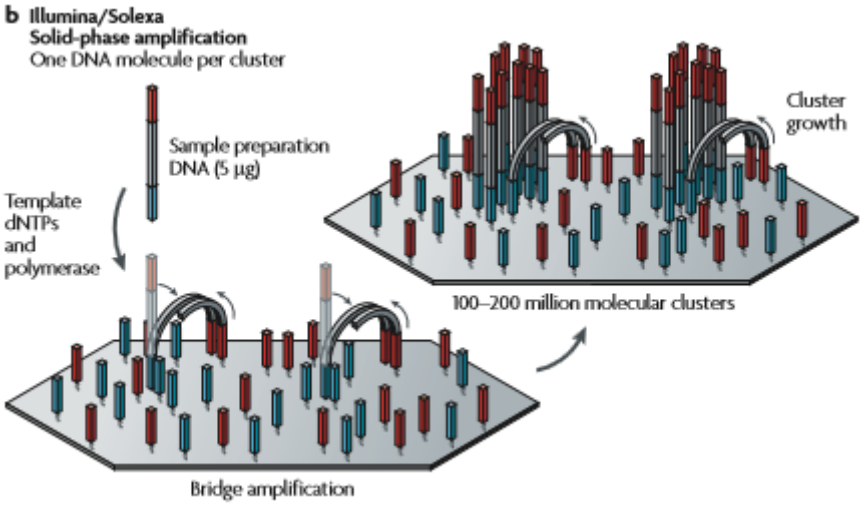


FIGURE 2-2 TEMPLATE PREPARATION IN ILLUMINA
Source: Metzker [11] (p.33)

The datasets used in this thesis are only **MiSeq** and **HiSeq** Illumina **paired end reads**. 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.

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

Resequencing on the other hand can be used to catalogue sequence variation. 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
- *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.

Both sequencing types can use a variety of starting materials including:

- Bacterial
- **Viral**
- **Phage**
- **Fungus**
- **BAC**s
- Fosmids
- Eukaryote genomic DNA
- Fragmented DNA

2.1.2 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 ideal result from an assembly is one continuous sequence equal to the target DNA, but it is not always the case. Repeats in the sequence can be one of the reasons that can make the ideal result difficult to achieve, and this will be elaborated upon in the subsection below named *Assembly challenges*. 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. The solution is suboptimal because it will only look at the next best fragment without considering what's best for the overall sequence. This process can be both time and resource consuming considering the amount and complexity of datasets researchers work with today. The algorithm is mostly abandoned today because it may for instance misassemble repeats.

Today, the two most used approaches for assemblers are Overlap – Layout - Consensus (which is used by programs such as Celera Assembler CABOG) and de Bruijn graph (which is used by programs such as ABySS and Velvet) [12]. The Overlap – Layout – Consensus is a well-established and powerful method, and the general idea behind OLC is quite simple. There are three steps to this approach:

1. Overlap
    a. 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 displayed in Figure 2-3 below:

```
Repeat 1:  GACCTACA
Repeat 2:    ACCTACAA
Repeat 3:      CCTACAAG
Repeat 4:        CTACAAGT
Read A:            TACAAGTT
Read B:              ACAAGTTA
Read C:                CAAGTTAG
Read X:            TACAAGTC
Read Y:              ACAAGTCC
Read Z:                CAAGTCCG
```

FIGURE 2-3 OLC STEP 1 - OVERLAP
Blue: Reads that covers repeated sections
Green: Reads that continue one repeated section
Purple: Reads that continues the same repeated section, but does not Overlap with the green reads A-C
Source: Schatz, Delcher and Salzberg [13]

2. Layout
    a. This is the step with the graph simplification. The reads are placed based on the alignment. By now, the overlap-step has finished aligning the sequence of reads which can be presented as a graph (Figure 2-4):

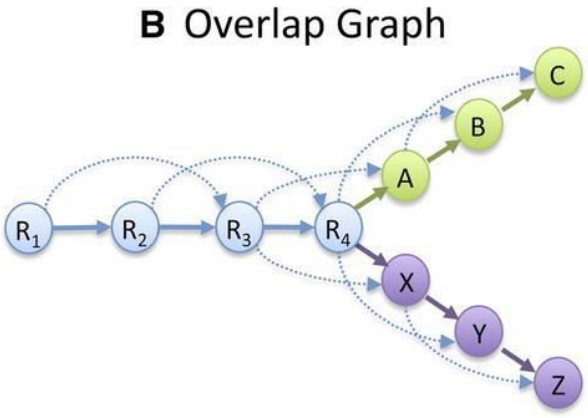


FIGURE 2-4 OLC STEP 2 - LAYOUT
Source: Schatz, Delcher and Salzberg [13]

    b. As seen from the 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 paths from R4, which can indicate that the reads R1-R2-R3-R4 might covers repeated parts in the original sequence as follows:
XXXXGACCTACAAGTTAGXXXXXGACCTACAAGTCCGXXXX
with X being unknown nucleotide sequences of unspecified length.

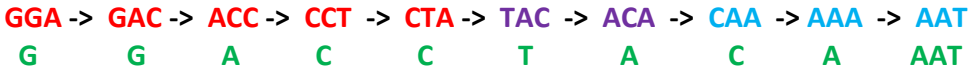
3. Consensus
    a. Step three; get consensus by joining all sequences of reads, merging overlaps that result in the final sequence.

De Bruijn graph is newer than the OLC method and although they both have essentially equivalent roles, they differ in the methods used to exploit the overlap information. While OLC constructs a read graph by assigning a link between two reads when they overlap by more than a cutoff length, de Bruijn graph constructs a k-mer graph that assigns a link between two k-mers when they are neighbors on the genome [12]. The drawback for de Bruijn graph 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 [4]. This graph theory algorithm 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.

As mentioned above, de Bruijn graph uses small k-mers which are found by iterating through the reads, base by base, and obtains all the k-mers available in the sequence. For instance, if we have the following reads: GGACCTACA and TACAAAT and uses k-mers of length 3, the k-words (colored and in bold) will be computed like this:

READ 1	**READ 2**
GGACCTACA	**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:



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 mentioned earlier in this chapter, the ideal result after performing an assembly is one continuous sequence which unfortunately is not the default case. There can be quite some assembly challenges to overcome for the sake of a continuous sequence. One paper that tries to discuss the challenges is *Genome assembly reborn: recent computational challenges* by Mihai Pop,[14] 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[15].

One of the problems regarding assembly process is genomic **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.

An assembler can incorrectly gauge the number of repeats by mis-joining reads originating from distinct repeat copies into one unit, or include extra copies of repeat, both which can be detected in the assembly with an unusual high or low density of reads. The assembler can also 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 misassembly. The probability of errors like the ones mentioned above can be reduced by for instance using sequencing technology which returns longer reads. This will, while assuming that reads have few sequencing errors, make it easier for assemblers to detect repeats and avoid misassemblies.

Considering the development of shorter reads and sequencing tools that generate several million reads, the complexity of an assembly, which depends on the number of reads, increases like never before. Let's think of this as a jigsaw puzzle again, with large stretches of sky, where it is possible to have thousands of pieces and not all pieces are unique. A puzzle like this with a thousand pieces would most likely be a lot harder than the same puzzle with just a hundred pieces. It might seem like the fewer reads the better, but even though longer (thus fewer) reads are easier to process, the shorter reads produce high coverage.

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.

2.1.3 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 criteria 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 criteria. 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 be weighted differently 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.

NUMBER OF CONTIGS OR SCAFFOLDS

This is defined as the total number of contigs (of size 200 bp or longer) or scaffolds (of size 500 bp or longer) in an assembly. In general, the fewer and longer the contigs/scaffolds are, the better it is. That is of course while assuming that the contigs/scaffolds are assembled correctly, which unfortunately is not true in all cases. This is where other features such as for instance the coverage, which tells the reliability of each nucleotide base position, or the number of misassemblies, might clarify the correctness of the contigs.

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 [8]. This thesis will use N50 values which mean that the sum of lengths of sequences greater than or equal in length to s is greater than or equal to 50% of the genome being assembled.

NAX

NAx of an assembly is the same as Nx except that it is where the lengths of aligned blocks are counted instead of contig lengths. I.e., if a contig has a misassembly with respect to the reference, the contig is broken into smaller pieces. It is also referred to as corrected N50, but the term used in this thesis will be NA50.

THE NUMBER OF MISASSEMBLIES

Misassemblies is characterized as the number of relocations, translocations and inversions affecting, in our case, at least 1000 bp, which is determined by comparison to the reference genome. Few misassemblies indicate that the assembled contigs/scaffolds are correct and it is therefore desired to have as few misassemblies as possible.

THE NUMBER OF LOCAL MISASSEMBLIES

Local misassemblies is defined as errors such as misjoins where the left and right pieces map onto the reference genome to distinct locations that are more than 1000 bp apart, or that overlap by more than 1000 bp. Just as with "global" misassemblies (relocations, translocations and inversions), the number of local misassemblies can be part of several features used to determine the correctness of assembled contigs/scaffolds and the fewer local misassemblies the better it is for an assembly.

THE NUMBER OF UNALIGNED CONTIGS/SCAFFOLDS

Unaligned contigs/scaffolds are defined as contigs/scaffolds that have no alignment (even partially) to the reference sequence at all. This should be as close to zero as possible because unaligned contigs/scaffolds indicate the errors.

GENOME FRACTION

Genome fraction 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.

DUPLICATION RATIO

The duplication ratio states 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.

NUMBER OF GENES

The number of complete genes in an assembly can be computed if an annotated list of genes positions in the reference genome is provided. At higher levels of coverage, if the number of contigs/scaffolds decreases and approaches the approximate number of genes then the quality of the assembly can be decided with more confidence.

2.1.4 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 [1] As you can see in the Figure 2-5 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.

Report Example

E. coli single-cell assemblies

Genome: E. coli, single-cell
4 639 675 bp, G+C content: 50.79 %
4324 genes, 884 operons

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

Extended report worst.....best

Statistics without reference	SPAdes-2.4	IDBA-UD	Velvet-SC	E+V-SC
# contigs	277	250	519	344
Largest contig	269 177	224 018	121 367	132 865
Total length	4 877 521	4 791 744	4 526 656	4 540 286
N50	106 927	96 947	20 445	33 616
Misassemblies				
# misassemblies	2	11	2	2
Misassembled contigs length	26 551	70 953	22 359	23 485
Mismatches				
# mismatches per 100 kbp	5.27	3.85	1.75	2.14
# indels per 100 kbp	0.79	0.27	0.94	0.73
# N's per 100 kbp	4.860	0	0	0
Genome statistics				
Genome fraction (%)	95.622	94.721	91.249	91.488
Duplication ratio	1.004	1.001	1.002	1.001
# genes	4047 + 98 part	4021 + 70 part	3626 + 277 part	3759 + 152 part
# operons	809 + 48 part	802 + 37 part	650 + 155 part	722 + 84 part
NGA50	110 539	96 947	19 791	32 051
Predicted genes				
# predicted genes (unique)	4580	4531	4459	4367
# predicted genes (>= 0 bp)	4669	4533	4459	4367
# predicted genes (>= 300 bp)	3804	3751	3694	3661
# predicted genes (>= 1500 bp)	560	559	515	524
# predicted genes (>= 3000 bp)	48	49	39	44

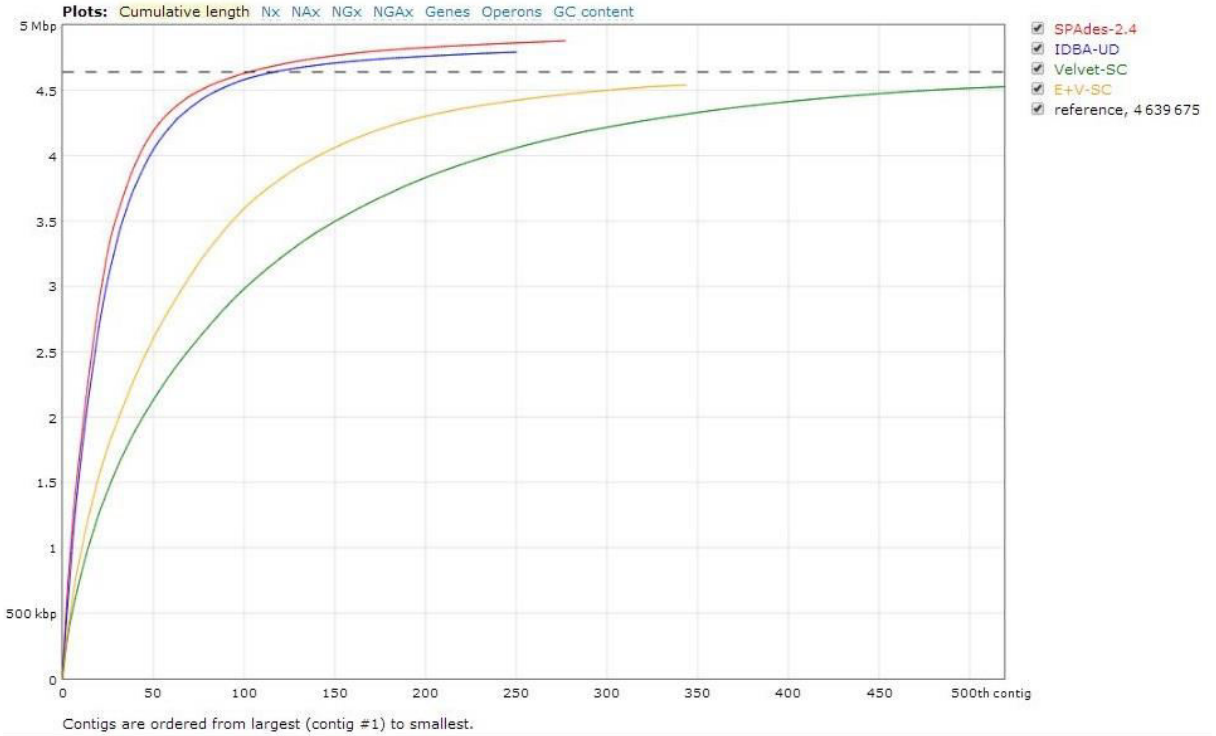


FIGURE 2-5 QUAST REPORT EXAMPLE
Source: http://QUAST.bioinf.spbau.ru/

2.2 REPRODUCABILITY

An experiment is reproducible until another laboratory tries to repeat it.
Alexander Kohn [16]

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. [17]

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 scientists 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. [18] has 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.

2.3 REUSABILITY

Good programmers know what to write. Great ones know what to rewrite (and reuse).
Eric S. Raymond [19]

The notion of reusability is, as stated by Prieto-Diaz in *Status Report: Software reusability* [20], 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.

2.4 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], 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.

[1] https://wiki.galaxyproject.org/
[2] http://insilico.hpc.uio.no:24688/history/list_published

2.4.1 GALAXY OBJECTS

Galaxy objects (Figure 2-6) are, in general, anything that can be saved, persisted and shared. Below is a list of galaxy objects that users may encounter:

FIGURE 2-6 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 (Figure 2-7) with anyone, whether they have a Galaxy-account or not.

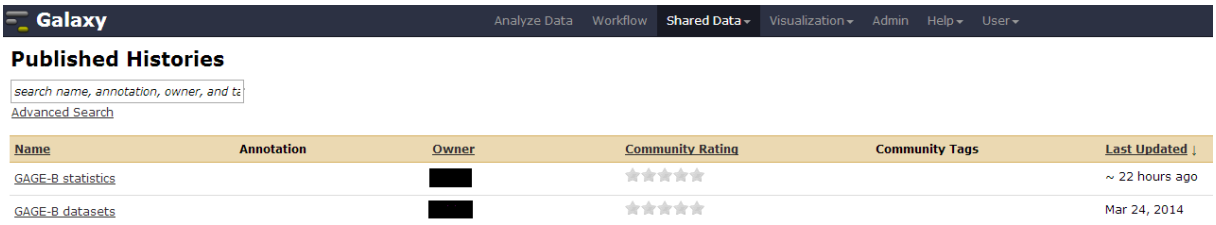


FIGURE 2-7 AN EXAMPLE OF SHARED OR PUBLISHED HISTORIES IN GALAXY

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 represents 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 2-8 and Figure 2-9 below shows an example of creation and running of a workflow in Galaxy.

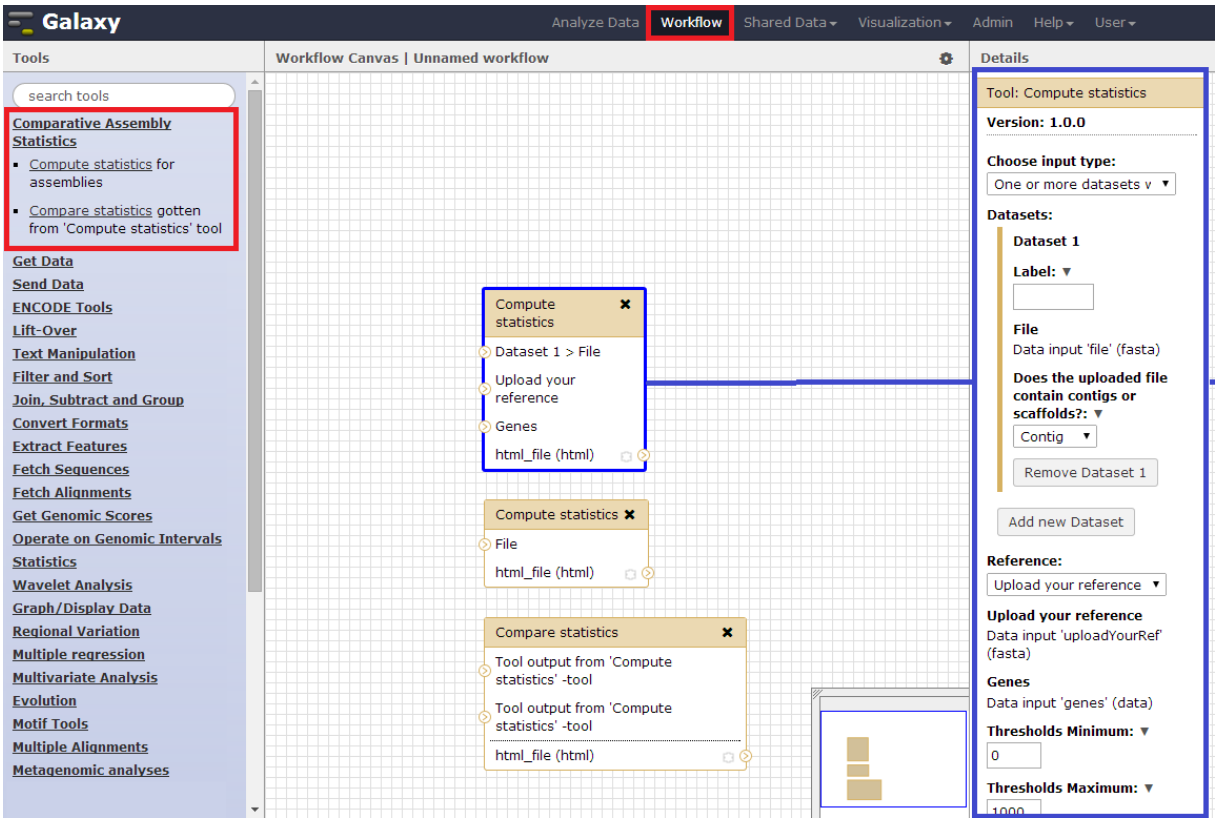


FIGURE 2-8 EXAMPLE OF CREATION OF A WORKFLOW

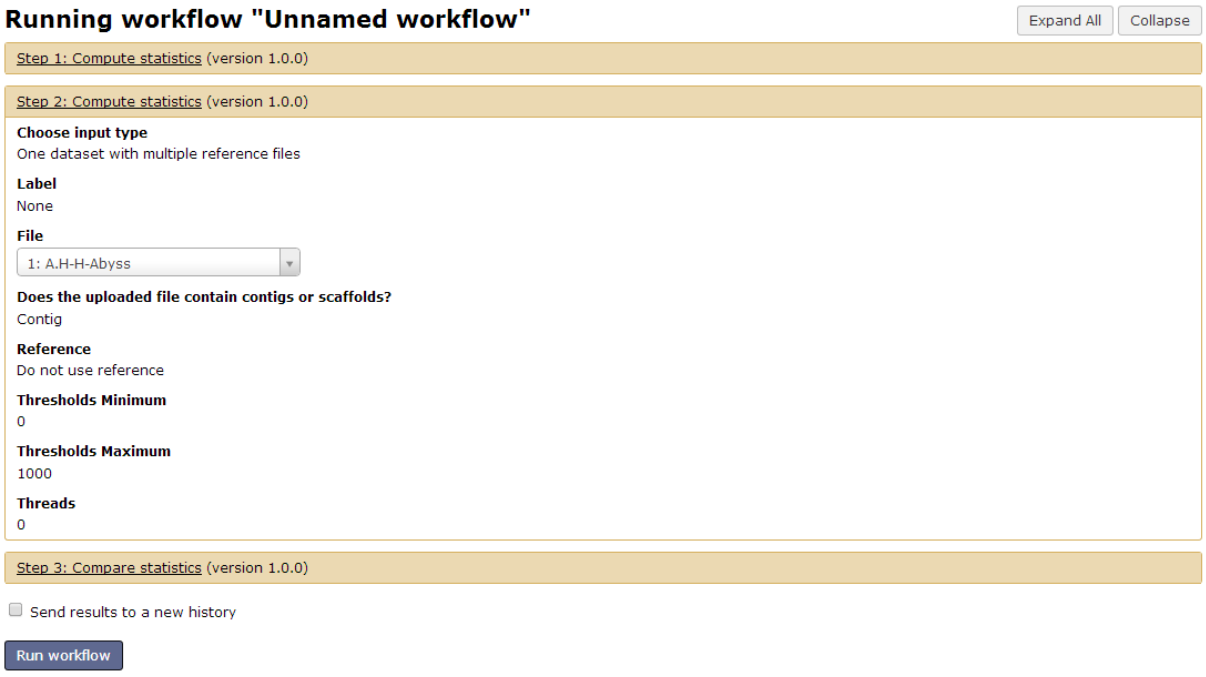


FIGURE 2-9 EXAMPLE OF RUNNING THE WORKFLOW FROM FIGURE 2-8

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.

2.4.2 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 and data types, as well as exported Galaxy workflows. It allows administrators 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.

CHAPTER 3 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.

3.1 DATASETS

There are three types of datasets used in this thesis, read data (**Error! Reference source not found.** and **Error! Reference source not found.**), assemblies and reference genomes (**Error! Reference source not found.**). 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.

3.1.1 READ DATA

TABLE 3-1 SPECIES, SEQUENCING TECHNOLOGY AND SIZE
Source: http://ccb.jhu.edu/gage_b/datasets/index.html

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

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 [21] or, as the GAGE-B researchers have done, by removing adapter sequences and performing q10 quality trimming using the ea-utils package.

TABLE 3-2 READ TYPE USED FOR EACH ASSEMBLER ON *VIBRIO CHOLERAE*

	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

3.1.2 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's webpage and as history elements from a list of published histories[2] or a published page[3] in Galaxy.

3.1.3 REFERENCE GENOMES

The reference genome and gene file used while trying to reproduce the GAGE-B assemblies was *Vibrio cholerae O1 biovar eltor str. 16961* (NC_002505 and NC_002506).

The reference genome and the gene files used for the assessment of the Galaxy tool were 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, size and RefSeq accession ID are shown in **Error! Reference source not found.** below.

TABLE 3-3 REFERENCE GENOME FOR EACHDATASET
Source: http://ccb.jhu.edu/gage_b/datasets/index.html

Reference	Type	Size (kB/MB)	RefSeq
Aeromonas hydrophila ATCC 7966	Chromosome 1	4.6 MB	NC_008570
Bacillus cereus ATCC 10987	Chromosome 1	5.1 MB	NC_003909
	Plasmid pBc10987	206 kB	NC_005707
Bacteroides fragilis 638R	Chromosome 1	5.2 MB	NC_016776
Mycobacterium abscessus	Chromosome 1	4.9 MB	NC_010397
	Plasmid 1	23.2 kB	NC_010394
Rhodobacter sphaeroides 2.4.1	Chromosome 1	3.1 MB	NC_007493
	Chromosome 2	934 kB	NC_007494
	Plasmid A	113 kB	NC_009007
	Plasmid B	113 kB	NC_007488
	Plasmid C	104 kB	NC_007489
	Plasmid D	100 kB	NC_007490

[2] http://insilico.hpc.uio.no:24688/history/list_published
[3] http://insilico.hpc.uio.no:24688/u/sabba/p/gage-b-datasets-and-statistics

	Plasmid E	36.8 kB	NC_009008
Staphylococcus aureus SA300_TCH1516	Chromosome 1	2.8 MB	NC_010079
	Plasmid pUSA300HOUMR	26.9 kB	NC_010063
	Plasmid pUSA01-HOU	3.2 kB	NC_012417
Vibrio cholerae O1 biovar eltor str. 16961	Chromosome 1	2.9 MB	NC_002505
	Chromosome 2	1.0 MB	NC_002506
Xanthomonas axonopodis pv. Citrumelo	Chromosome 1	4.8 MB	NC_016010

3.2 SOFTWARE

This section covers all the software used in this thesis for both reproduction of GAGE-B results and 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.

3.2.1 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[22]

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

Many versions have been released since the assemblies were computed 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

3.2.2 CABOG

CABOG [23] 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 are reconstructed from fragmentary data produced by WGS sequencing.

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

3.2.3 MIRA

Mimicking Intelligent Read Assembly (MIRA) [24] 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 (same as in GAGE-B) which can be downloaded from http://www.chevreux.org/project_mira.html

3.2.4 MASURCA

MaSuRCA (MSRCA) [25] 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 (same as GAGE-B) 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/

3.2.5 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**. [26]

The version used in this thesis is the same as in GAGE-B, 0.9.34 (released August 23, 2012) while the current release is version 0.10.13 (released January 17, 2014) which can be downloaded from https://github.com/jts/sga/releases.

3.2.6 SOAPDENOVO2 + GAPCLOSER

Short Oligonucleotide Analysis Package denovo (SOAPdenovo) [27] is a short-read assembly method (especially 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 (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 emerge 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.

3.2.7 SPADES

St. Petersburg genome assembler (SPAdes) is a genome assembler designed for bacterial data. It works with Ion Torrent, PacBio and Illumina paired-end, mate-pairs and single reads. [28]

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/

3.2.8 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 (Illumina) 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. [4]

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_1.2.08.tgz
https://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz

3.2.9 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. Python 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. [29]

The following Python modules and libraries have been implemented:

OS MODULE

This Python 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.[30] 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.

TIME MODULE

This Python module provides various time-related functions. [31] It is used to get current date and time in a string format.

ZIPFILE MODULE

This Python module provides tools to create, read, write, append, and list a ZIP file. [32]

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 [33]

REPORTLAB

This is the ReportLab PDF Toolkit [34]. It allows rapid creation of rich PDF documents, and also creation of charts in a variety of bitmap and vector formats. It consists of several packages where the two used in this thesis are *pdfgen* and *rl_config*.

The pdfgen package is the lowest level interface for generating PDF documents. The interface object used in this thesis for "painting" a document onto a sequence of pages is the pdfgen canvas. 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. [35]

COLLECTIONS

This Python module provides alternatives to Python's built-in containers, dict, list, set, and tuple by implementing specialized container data types such as OrderedDict which is a dict subclass that remembers the order entries were added. [36]

3.2.10 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. [37]

3.2.11 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. [38]

3.2.12 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. [39]

3.2.13 JSON

JavaScript Object Notation (json) is a lightweight data interchange format based on a subset of JavaScript syntax. [40] It can be used to load an external json file and to dump the content to a new file.

CHAPTER 4 METHODS

This chapter covers a description of the tool and implementation. Read type (raw or cleaned) and assemblers used in the reproduction of GAGE-B results is listed in Table A 1. A 'recipe' used for each assembler can be viewed in Table A 2 or downloaded as a text file from https://github.com/subway/masterthesis/tree/master/Supplementary%20Material

4.1 REPRODUCING THE GAGE-B RESULTS

All the runs were performed in a Linux based environment with the programs from Section 3.2. Each assembly was computed for both MiSeq and HiSeq data where some reads were trimmed/cleaned and others were raw as described in the GAGE-B papers supplementary file and **Error! Reference source not found.**. A common set of data cleaning steps were performed by the GAGE-B authors 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. 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 Table A 1.

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 species (thus 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.

4.2 NEW GALAXY TOOLS FOR ASSEMBLY EVALUATION

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.

4.2.1 PUBLISHED GALAXY OBJECTS

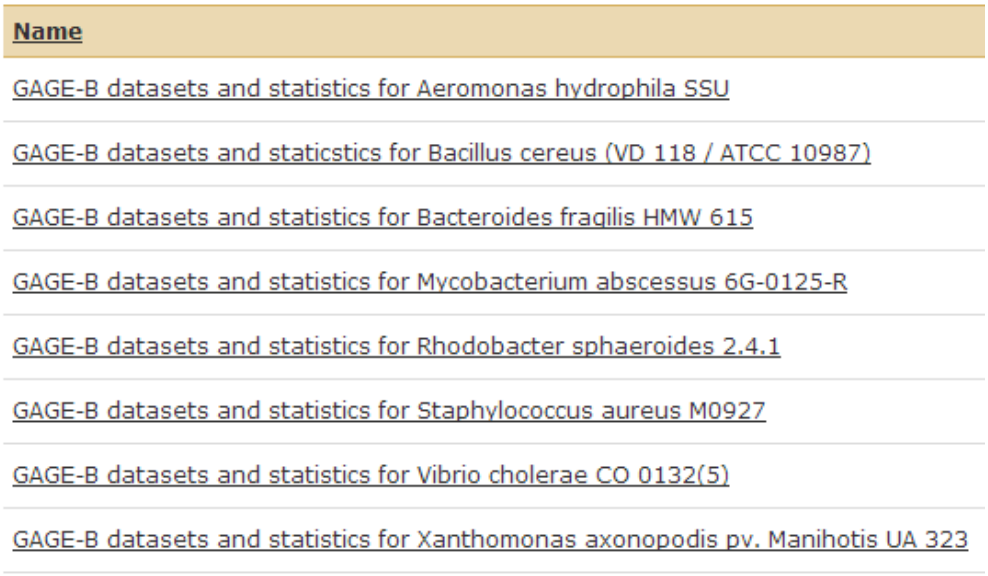


FIGURE 4-1 LIST OF PUBLISHED GALAXY HISTORIES

All the GAGE.B assemblies were used as input for several histories that were later published (Figure 4-1). 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, read type (HiSeq/MiSeq), assembler and data type (contig/scaffold). All assembler names 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
http://insilico.hpc.uio.no:24688/u/sabba/p/GAGE-B-datasets-and-statistics

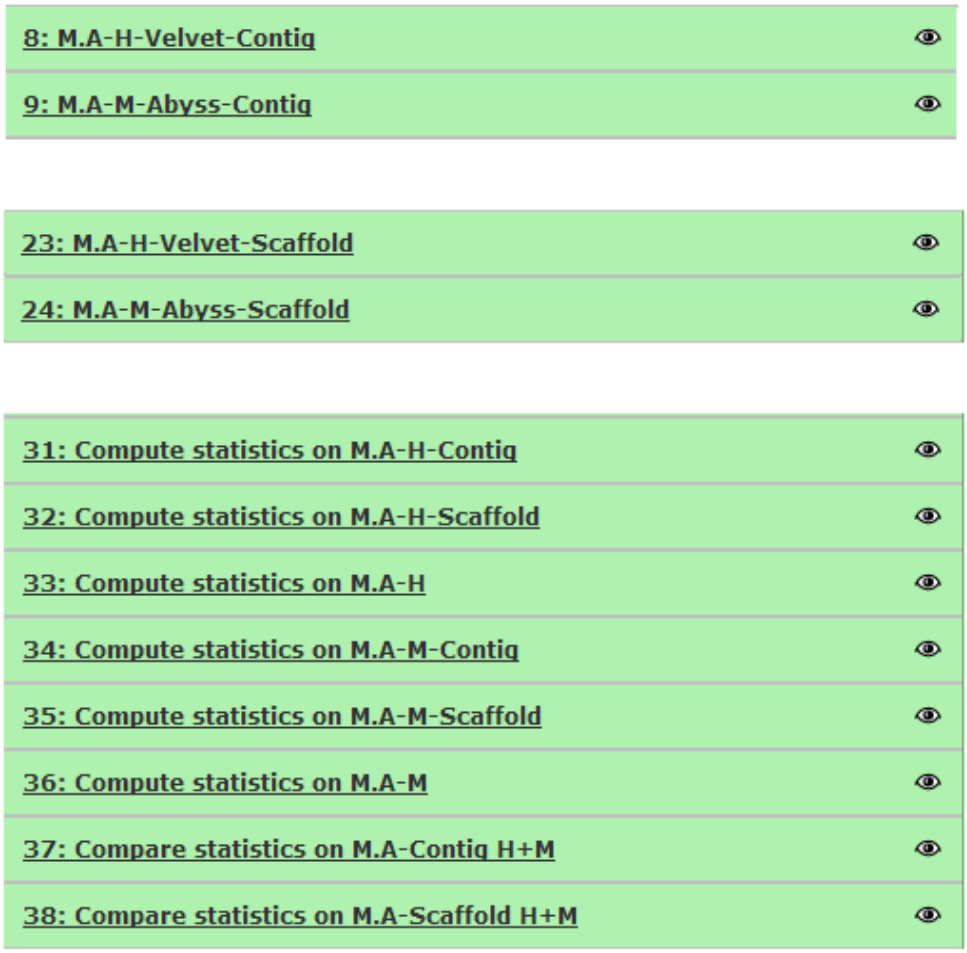


FIGURE 4-2 THE NAMING OF DATASETS FOR *MYCOBACTERIUM ABSCESSUS 6G-0125-R*

4.2.2 IMPLEMENTATION AND TESTING OF GALAXY TOOLS

The Galaxy instance used in this thesis is available at http://insilico.hpc.uio.no:24688 and the code can be downloaded from https://github.com/subway/Galaxy-Distribution for those more interested in the implementation. The structure of both the Galaxy instance and the tool for this thesis are shown in Figure 4-3. Note that only altered folder/files are included. A separation line indicate that the instance is a folder while an instance without the separation line is a file.

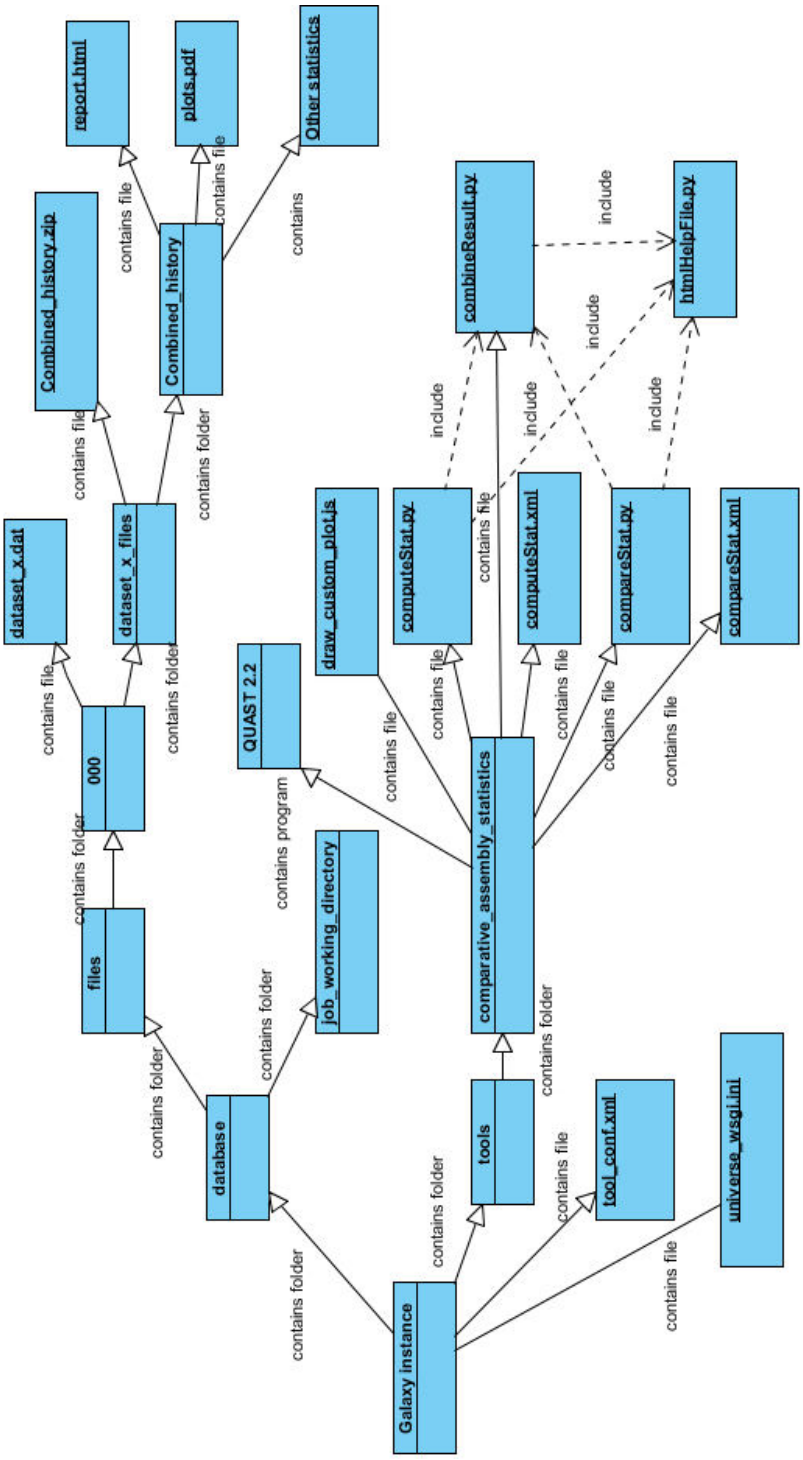


FIGURE 4-3 STRUCTURE OF THE GALAXY INSTANCE

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. These modules need to be importable if Python is user compiled. Galaxy requires a few things in addition 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, insilico.hpc.uio.no. 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 (Figure 4-4). 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`

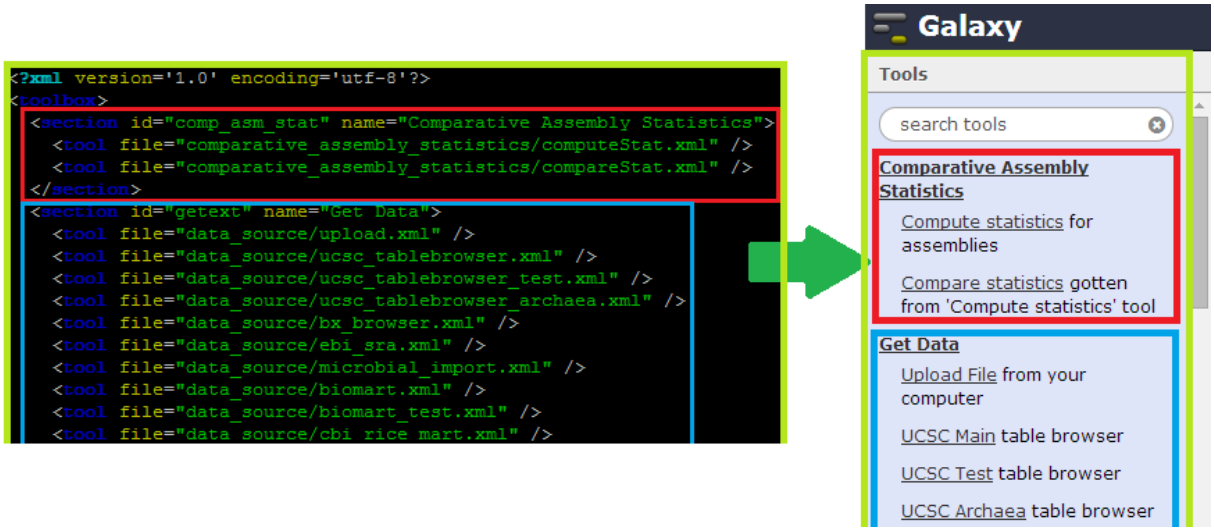


FIGURE 4-4 THE TOOL_CONF.XML FILE AND THE TOOL MENU IN GALAXY

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 and 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 evaluate assemblies by computing 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 accessed from github[4]. 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 run 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.

[4] https://github.com/subway/Galaxy-Distribution

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 two functions that take advantage of Google Charts to create interactive plots used in the report.html file in the output of both tools. The first function (drawColumnChart) creates bar charts and the second function (drawScatterPlot) creates scatterplots. A third function (toggleImg) is used to toggle the plot, i.e. hiding the plot at the

users command when displayed while the last function (saveImg) is used to create png-versions of the plots current state. 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 dropdown list 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 dropdown lists, 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 as well as the Python module os to validate the different inputs thresholds values

DATABASE FOLDER.

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 what's 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 unique, numeric value which is incremented by 1 for each entry. This is worth noting 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.

4.2.3 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 gene file 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 4-5.

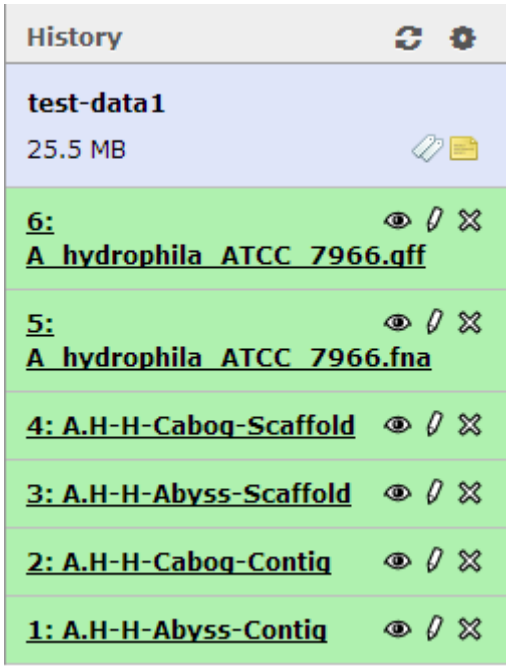


FIGURE 4-5 A HISTORY WITH TEST-DATA USED IN THE USER MANUAL

COMPUTE STATISTICS

This tool uses at least one assembly (reference genome and gene file is optional) to compute statistics as QUAST results with improved visualization features. Figure 4-6 displays the default start page for this tool. The output from this tool can be used as input for "Compare statistics"-tool where 2 or more results can be merged into one.

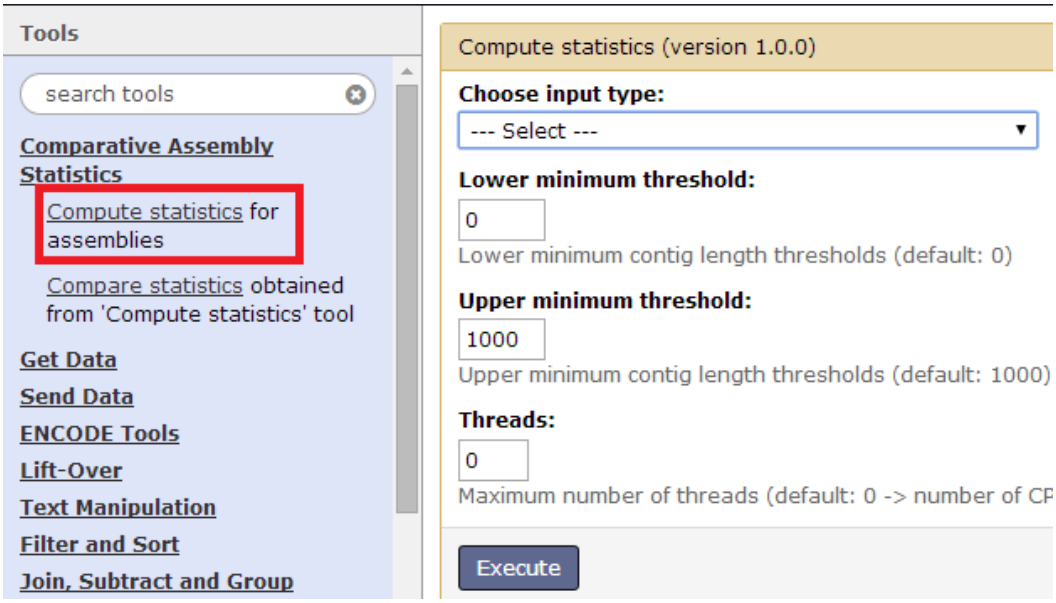


FIGURE 4-6 DEFAULT STARTPAGE FOR THE "COMPUTE STATISTICS"-TOOL

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 4-7.

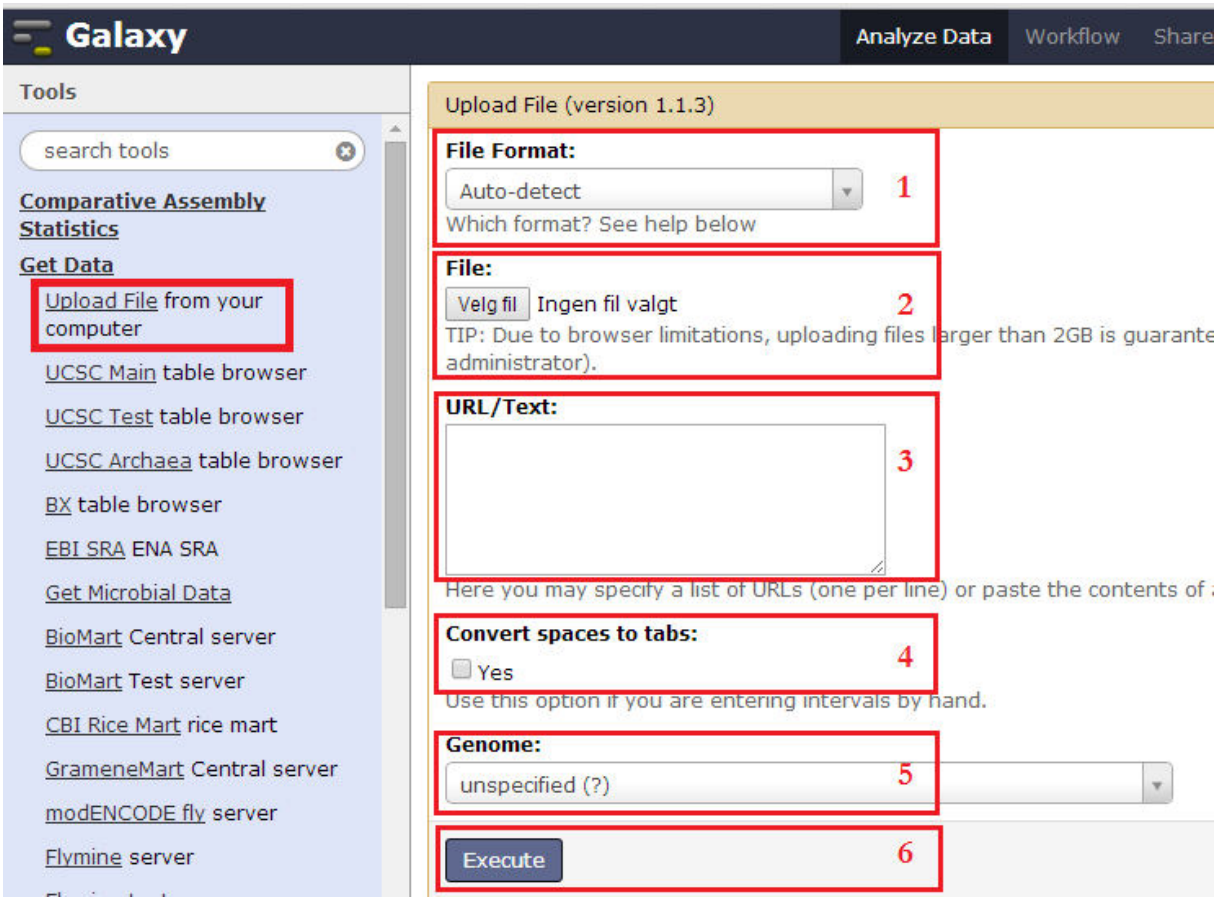


FIGURE 4-7 BLANK PAGE FOR THE TOOL "UPLOAD FILE"

1. File format - let the toll auto-detect the 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 a tab separated file
5. Genome - optional to specify which genome the file correspond to
6. Execute - press this to start the file upload to the 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 4-8 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 history elements 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.

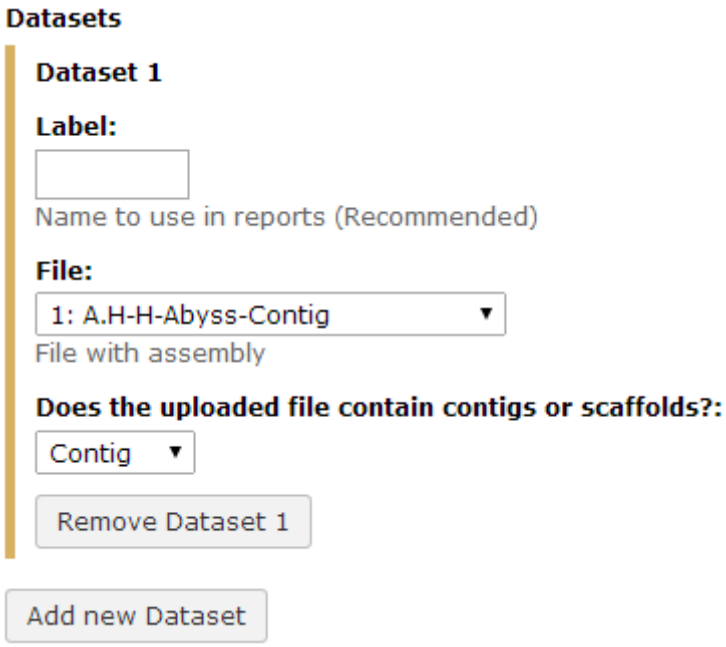


FIGURE 4-8 SCREENSHOT OF NEW FIELDS AVAILABLE AFTER PRESSING "ADD NEW DATASET"

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:

a) 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.

b) Use one built in reference genome (Figure 4-9) with optional use of corresponding gene file

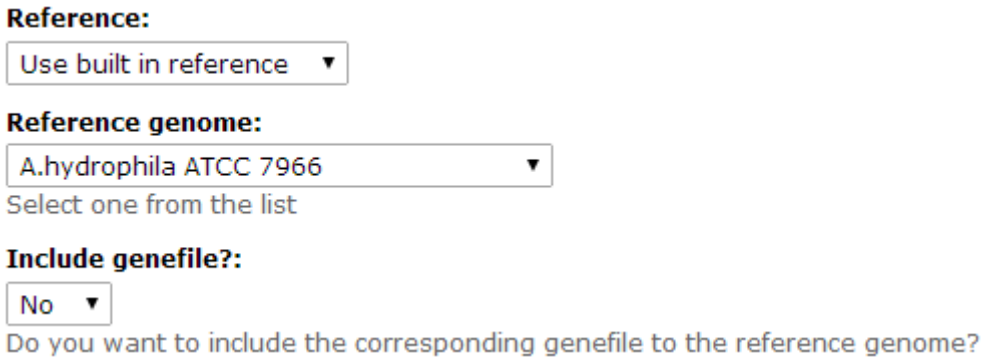


FIGURE 4-9 SCREENSHOT OF THE OPTION USE "BUILT IN REFERENCE"

c) Upload another reference genome (Figure 4-10) with optional upload of corresponding gene file. This option allows the user to upload one reference file containing a single reference genome.

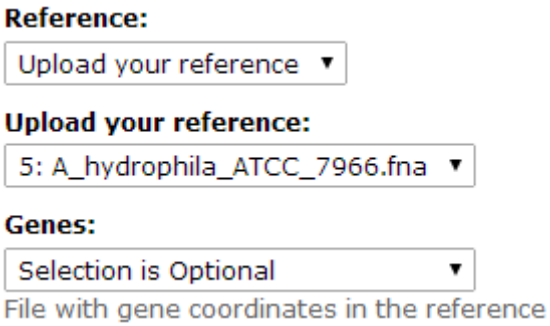


FIGURE 4-10 SCREENSHOT OF THE OPTION "UPLOAD YOUR REFERENCE"

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 4-8 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 history elements 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 most one assembly with at least one reference genome. The reference type is set by the option "reference" where the user can choose to:

a) Use at least one built in reference genome (default, Figure 4-11) with optional use of corresponding gene file.

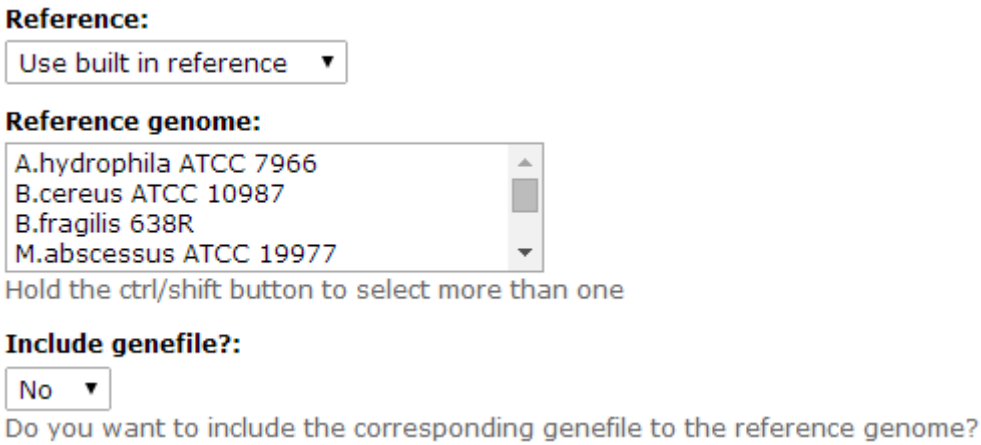


FIGURE 4-11 SCREENSHOT OF THE OPTION "USE BUILT IN REFERENCE"

b) Upload another reference genome (Figure 4-12) 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.

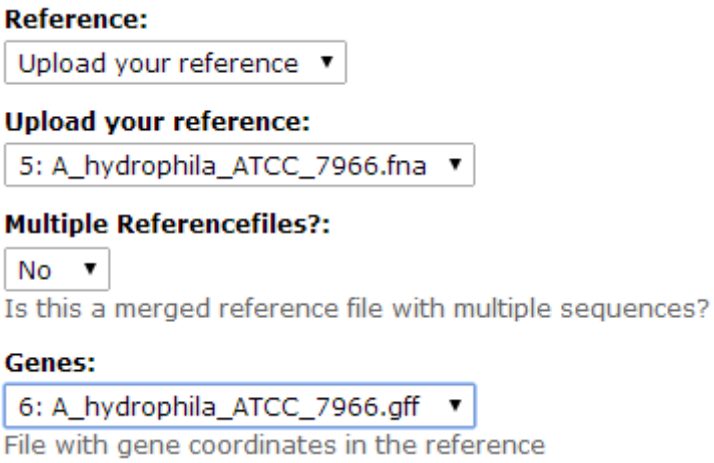


FIGURE 4-12 SCREENSHOT OF THE OPTION "UPLOAD YOUR REFERENCE"

Last, but not least, some QUAST specific parameters (Figure 4-13) such as minimum/maximum thresholds and the number of threads can be customized regardless of the option chosen in "Choose input type".

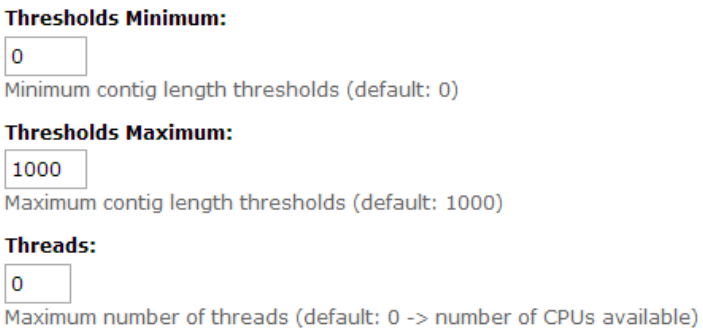


FIGURE 4-13 SCREENSHOT OF QUAST SPESIFIC PARAMETERS WITH THEIR DEFAULT VALUES

The lower and upper, minimum threshold parameters are used in # contigs ≥ x and total length (≥ x) metrics and the default value is [0, 1000]. 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 (Figure 4-14) 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 .

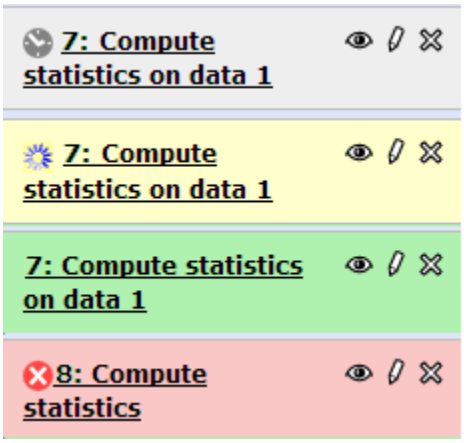


FIGURE 4-14 STAGES OF TOOL EXECUTION

VIEW RESULTS

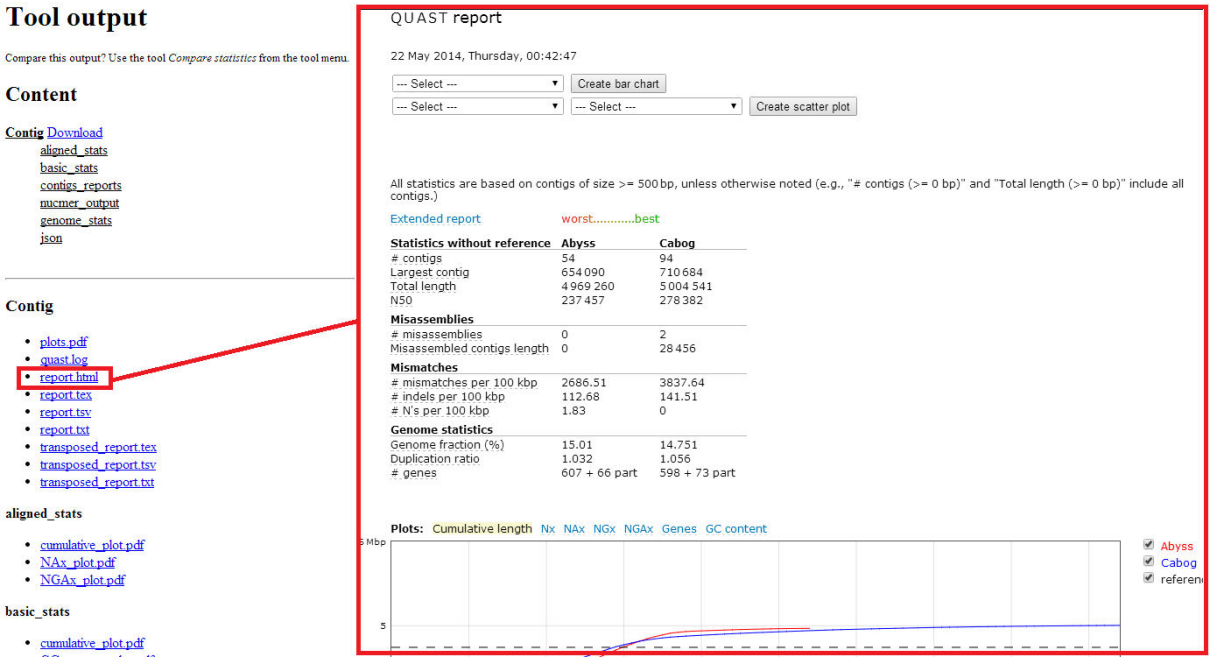


FIGURE 4-15 EXAMPLE OF TOOL OUTPUT AND IN BROWSER VIEW OF REPORT.HTML

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 (Figure 4-15) 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 4-16. 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 (Figure 4-16 "NEW TABLE"). 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.

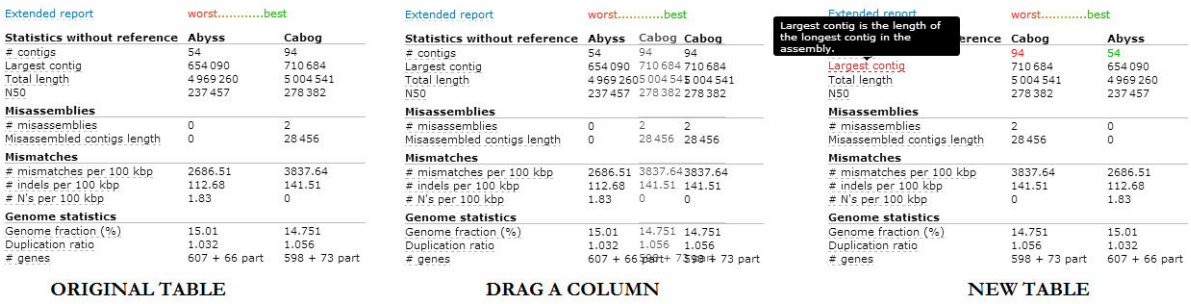


FIGURE 4-16 EXAMPLE OF AN INTERACTIVE TABLE IN REPORT.HTML

This file provide the user with at least 3 (cumulative length, Nx, GC content) and up to 10 QUAST generated, 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 (Figure 4-17).

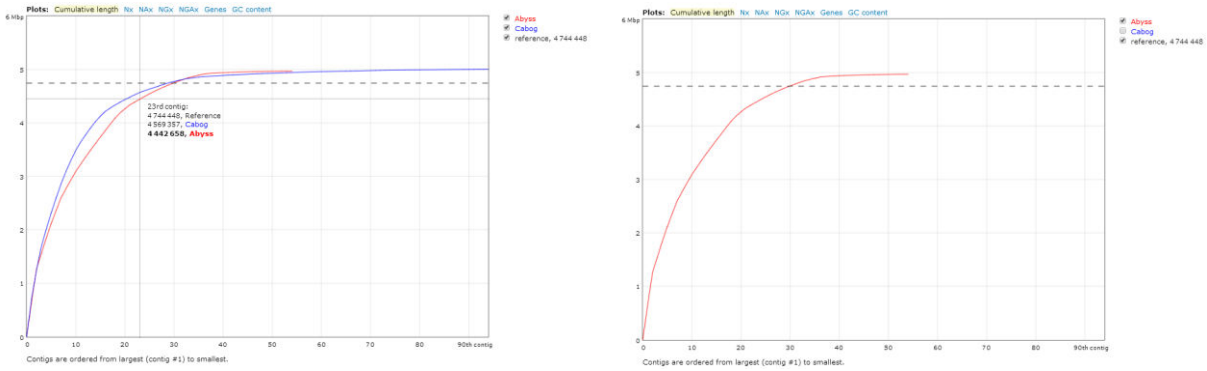


FIGURE 4-17 PREDRAWN PLOT IN REPORT.HTML
Left: Original plot (cumulative length) with mouse hovering over the graph
Right: Same plot, but CABOG is unchecked from the legend at the right side of the plot making a plot with only ABySS data

Each of these predrawn plots is mentioned in **Error! Reference source not found.** below with a short description retrieved from QUAST's user manual [26].

TABLE 4-1 DESCRIPTION OF PREDRAWN PLOTS IN REPORT.HTML

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 unaligned 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 unaligned 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	

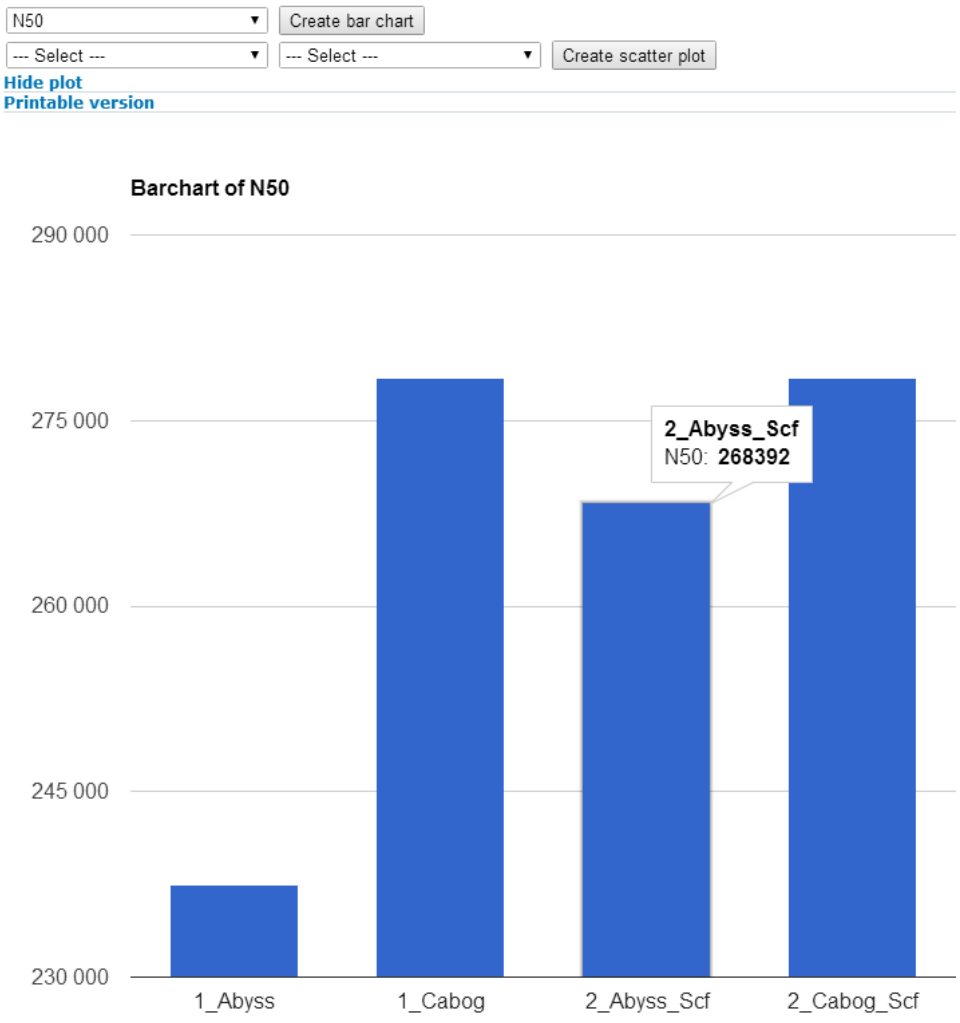


FIGURE 4-18 BAR CHART IN REPORT.HTML

The user can also create other plots with metrics of their own choice as either a bar chart for one metric (Figure 4-18) or as a scatterplot for two metrics (Figure 4-19). These plots are extra, interactive, visualization features implemented on top of QUAST's original features for this file (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. It is also possible to create an image of the dynamic plot by clicking "printable version" which opens the current state of the plot as an image in a new tab. The bar chart 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" or create an image by clicking on the blue "printable version". 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 4-19-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 4-19-Left plot). Note that only one point at a time can be selected and that selecting a point automatically deselect any previously selected point.

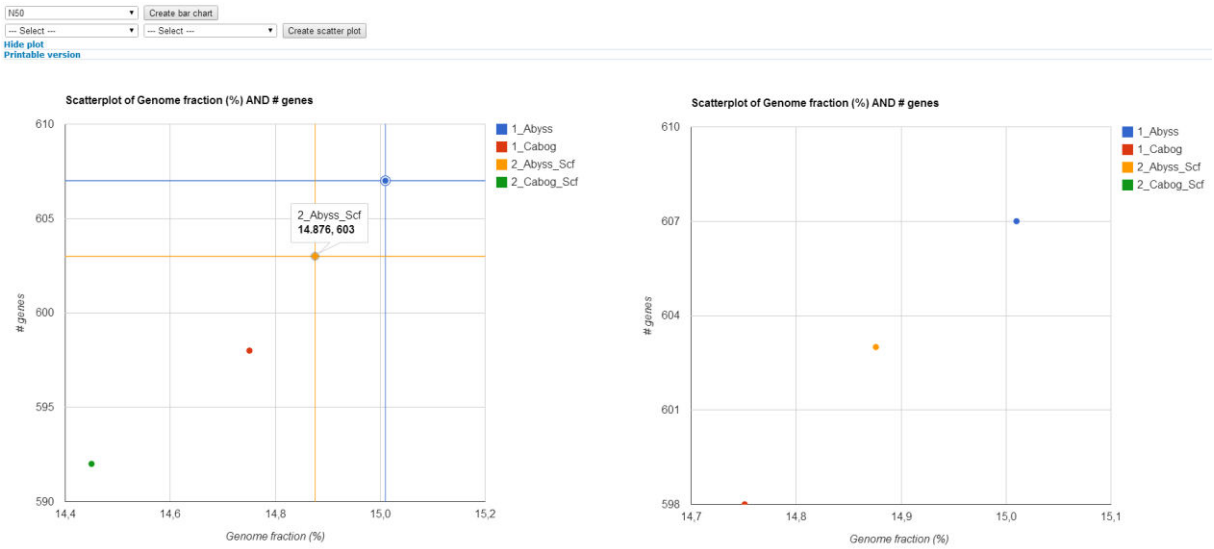


FIGURE 4-19 SCATTERPLOT IN REPORT.HTML
Left: Original plot showing selectable point (yellow) with a hovered point (blue)
Rigth: 2_Cabog_Scf (green) removed from displayed elements

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 versions 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) obtained from the "Compute statistics"-tool. Figure 4-20 displays the default start page for this tool.

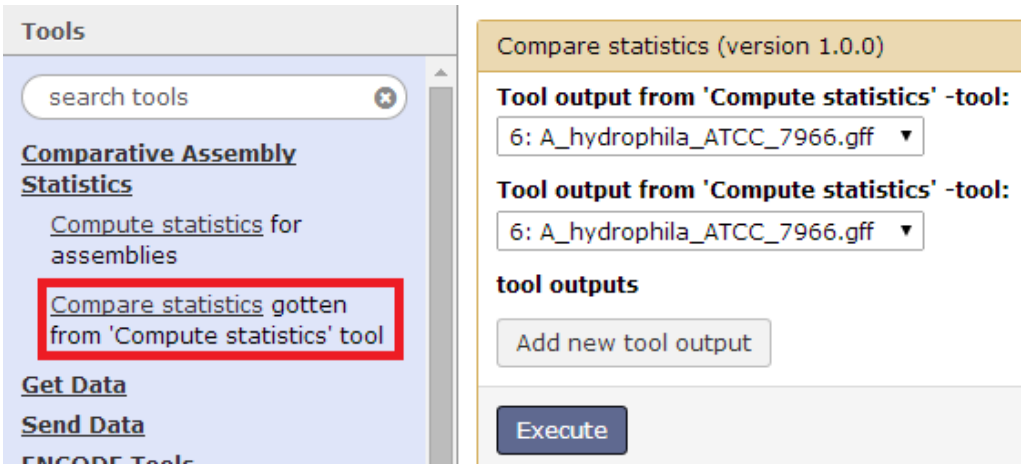


FIGURE 4-20 DEFAULT STARTPAGE FOR THE "COMPARE STATISTICS" TOOL

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

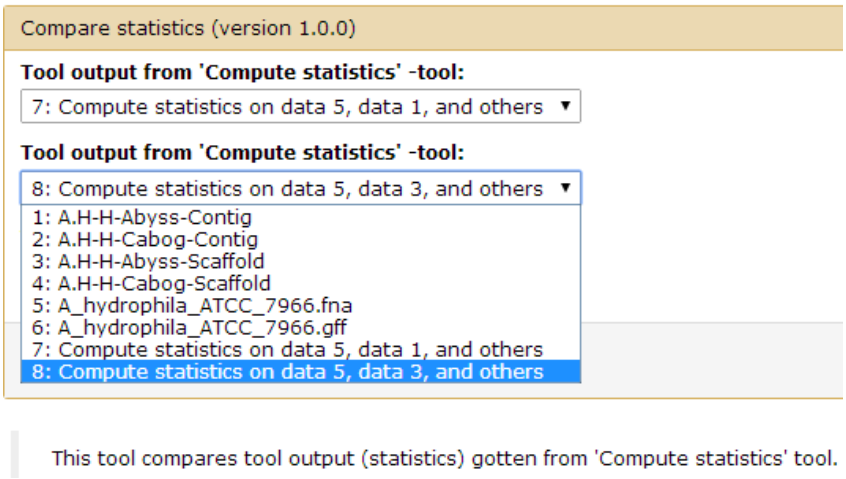


FIGURE 4-21 SCREENSHOT OF HOW TO CHOOSE INPUT DATA FOR "COMPARE STATISTIC"

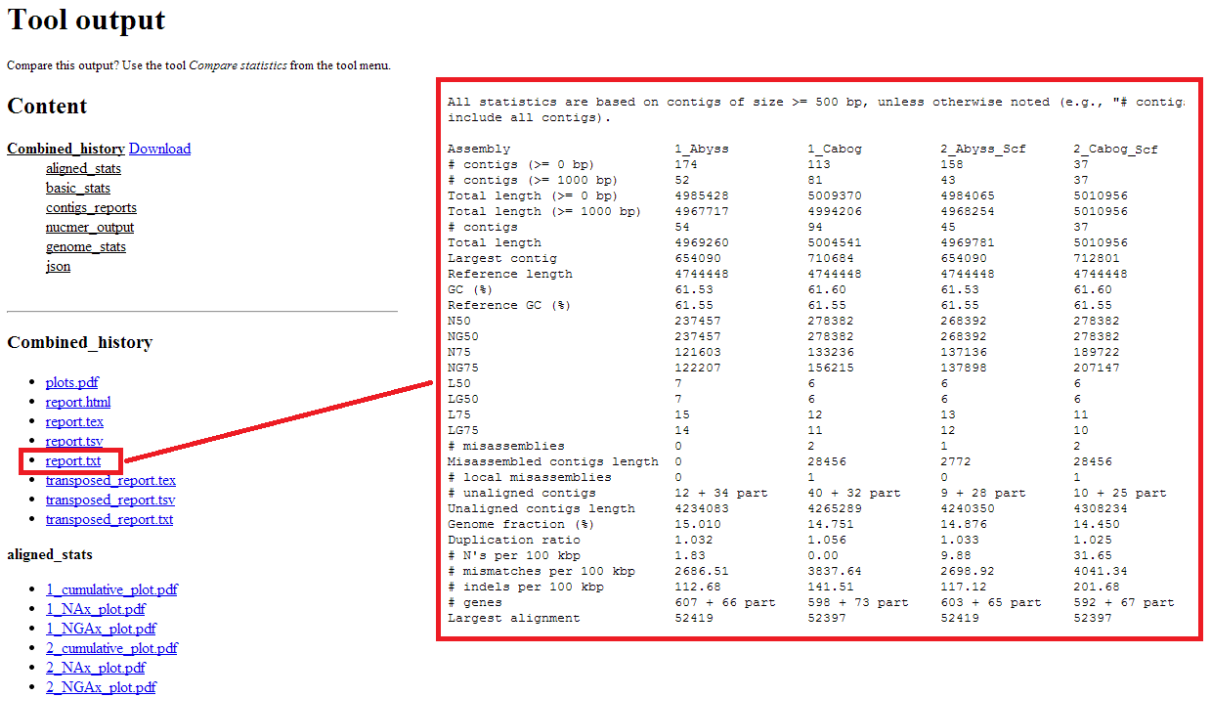
This tool requires that you choose input from dropdown lists with the history elements from the current history (Figure 4-21). Note that multiple inputs with the same history element will be omitted and that this tool requires at least two unique history elements. It will cancel the execution if two histories run with different lower/upper thresholds values are selected.

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



Tool output

Compare this output? Use the tool *Compare statistics* from the tool menu.

Content

Combined_history Download
- aligned_stats
- basic_stats
- contigs_reports
- nucmer_output
- genome_stats
- json

Combined_history

- plots.pdf
- report.html
- report.tex
- report.tsv
- report.txt
- transposed_report.tex
- transposed_report.tsv
- transposed_report.txt

aligned_stats

- 1_cumulative_plot.pdf
- 1_NAx_plot.pdf
- 1_NGAx_plot.pdf
- 2_cumulative_plot.pdf
- 2_NAx_plot.pdf
- 2_NGAx_plot.pdf

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contig include all contigs).

Assembly	1_Abyss	1_Cabog	2_Abyss_Scf	2_Cabog_Scf
# contigs (>= 0 bp)	174	113	158	37
# contigs (>= 1000 bp)	52	81	43	37
Total length (>= 0 bp)	4985428	5009370	4984065	5010956
Total length (>= 1000 bp)	4967717	4994206	4968254	5010956
# contigs	54	94	45	37
Total length	4969260	5004541	4969781	5010956
Largest contig	654090	710684	654090	712801
Reference length	4744448	4744448	4744448	4744448
GC (%)	61.53	61.60	61.53	61.60
Reference GC (%)	61.55	61.55	61.55	61.55
N50	237457	278382	268392	278382
NG50	237457	278382	268392	278382
N75	121603	133236	137136	189722
NG75	122207	156215	137898	207147
L50	7	6	6	6
LG50	7	6	6	6
L75	15	12	13	11
LG75	14	11	12	10
# misassemblies	0	2	1	2
Misassembled contigs length	0	28456	2772	28456
# local misassemblies	0	1	0	1
# unaligned contigs	12 + 34 part	40 + 32 part	9 + 28 part	10 + 25 part
Unaligned contigs length	4234083	4265289	4240350	4308234
Genome fraction (%)	15.010	14.751	14.876	14.450
Duplication ratio	1.032	1.056	1.033	1.025
# N's per 100 kbp	1.83	0.00	9.88	31.65
# mismatches per 100 kbp	2686.51	3837.64	2698.92	4041.34
# indels per 100 kbp	112.68	141.51	117.12	201.68
# genes	607 + 66 part	598 + 73 part	603 + 65 part	592 + 67 part
Largest alignment	52419	52397	52419	52397

FIGURE 4-22 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 (Figure 4-22) 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*

CHAPTER 5 RESULTS

The results computed for this thesis are divided into subsections covering whether it is inconsistent GAGE-B results, reproduction of GAGE-B results or reusability of GAGE-B datasets. Table B 1 to Table B 22 presents statistics on each *Vibrio cholerae* assembly (computed by QUAST) 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 (# misassemblies) affecting at least 1000 bp, which is determined by comparison to the reference genome
- The number of local errors such as misjoins (# local misassemblies) 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 unaligned contigs that have no alignment (even partially) to the reference sequence
- The fraction of the 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)

A detailed explanation of the metrics above was described in Section 2.1.3 while full QUAST generated statistics can be examined in Table S 1 Table S 22. All the metrics above are used for 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 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.

5.1 INCONSISTENT GAGE-B RESULTS

QUAST results generated from GAGE-B assemblies retrieved from their webpage[5] were subject to comparison against GAGE-B's supplementary material because the initial runs discovered quite a difference on various metrics. Since the GAGE-B authors do not mention QUAST options (for instance --gage) used to obtain their results in either the paper or the supplementary material , all QUAST runs were computed with default QUAST options except for a lower minimum threshold value of 200 for contig files and 500 for scaffold files. Example of QUAST commands for contig and scaffold files:

[5] http://ccb.jhu.edu/gage_b/genomeAssemblies/index.html

For assembly evaluation of contig files

```
quast.py –o [directory name] –R [reference] –G [gene file] \
--contig-threshold 200,1000 contig.fasta
```

For assembly evaluation of scaffold files

```
quast.py –o [directory name] –R [reference] –G [gene file] \
--contig-threshold 500,1000 --scaffold scaffold.fasta
```

The comparison of all contig and scaffold files were computed using the new Galaxy tools for assembly evaluation and the results are highlighted in Table B 23 and Table B 24.

The most consistent values are the number of contigs/scaffolds which is exactly the same except for 1-5bp difference for contig files computed by MIRA and SGA. The values for the N50 and NA50 metrics are partially equal, but for assemblies resulting in a difference, the worst results of 0.3-19.4 kb (N50) or 0.2-15.6 kb (NA50) were listed under supplementary materials. The only exception here is the scaffold file obtained from Velvet runs on MiSeq data where the value in the supplementary material (92.0kb) exceeds the assemblies NA50 (75.9kb) by 16.1kb. The number of misassemblies (inversion, relocation and translocation) is different for all assemblers on both contig and scaffold files. Each assembly has fewer misassemblies than the number listed in GAGE-B's supplementary material. The number of local misassemblies for both contig and scaffold files have only one third of the assembly values equal to the ones listed in GAGE-B's supplementary material. The number of unaligned contigs/scaffolds is also different on all assemblers except for in 5 HiSeq data (CABOG and SGA contigs/scaffolds and Velvet scaffold). Just like the number misassemblies, the genome fraction is different on all assemblers for both contig and scaffold files. GAGE-B's supplementary material has listed between 0.5-3.0% higher coverage of the genome compared to the released assemblies. Assuming that the size of the reference genome is 4 033 464bp, the difference equals to roughly 20-121kbp. The last metric to be compared is the duplication ratio. It is, when rounded up to 1 decimal, the same for all assemblers except for three cases: the contigs and scaffolds for ABySS with HiSeq data where the ratio is higher on the assemblies, and contigs for SGA MiSeq data where the ratio is lower than the assembly computed ratio.

Contigs computed by CABOG 7.0 assembly on HiSeq data were rerun on QUAST, this time with the --gage option, to check whether or not it could be the reason to all the differences mentioned below, but the results were still inconsistent. On top of that, not all metrics used in the paper were available, making it reasonable to assume that QUAST runs were performed without the --gage option.

The last thing worth mentioning is that the supplementary material displays different reference genome size for MiSeq and HiSeq data. Assemblies for *Vibrio cholerae* with HiSeq data has a reference genome size of 4 033 464bp (the same as in the downloaded assembly files), while the genome size for assemblies with MiSeq data is 4 967 469bp.

5.2 REPRODUCING GAGE-B RESULTS

All GAGE-B results were tentatively reproduced using GAGE-B supplied reads and recipe for assemblies unless otherwise noted in Table A 1 and Table A 2. 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 access permissions 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 species *Vibrio cholerae*. 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.

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. The results from assemblies performed on *Vibrio cholerae* are described below.

5.2.1 ASSEMBLER SPECIFIC COMPARISON TO GAGE-B RESULTS

ABySS and SGA ran with errors and are both excluded from further comparison.

CABOG 7.0

For results computed on HiSeq data using CABOG 7.0, GAGE-B's contigs were better on all metrics, except for unaligned contigs (equal) and duplication ratios (0.001 higher). The scaffolds had worse results on all metrics except for unaligned scaffolds (equal), genome fraction (0.308 % better), and duplication ratio (0.011 better). Results from MiSeq data (contigs and scaffolds) were, for some unknown reason, surprisingly bad with a coverage of only 7.639% of the genome fraction compared to GAGE-B's 96.968%. The number of contigs/scaffolds (241 vs.188) and the number of global/local misassemblies are the only metrics with better values than the GAGE-B results. The biggest difference is 94 more unaligned contigs/scaffolds on the reproduced results. With this said, the assembly on MiSeq data has the most significant differences on all the other metrics with a highly disturbing result compared to GAGE-B.

MIRA 3.4.0

The comparison for the MIRA assembler is based on contigs only because it does not create scaffolds. Even though the assembly computed with the HiSeq data have covered more than 90% of the genome, the overall values for all the other metrics (except duplication ratio) are worse than the listed values for GAGE-B results. The contigs computed with MiSeq data is closer to GAGE-B results and actually better considering the number of contigs, errors (global/local misassemblies and unaligned contigs) and duplication ratio.

MASURCA 1.8.3

The recipe written by the GAGE-B authors stated that the k-value used in assemblies with MaSuRCA was 89 *and* 99 for HiSeq data, leaving MiSeq data without a k-value. The runs with MaSuRCA were therefore computed with both k-values for both HiSeq and MiSeq data. The assemblies performed on MiSeq data were unfortunately computed unsuccessfully with gapclose error as described in Table A 3. Which result to use in the comparison was determined by looking at the best values for N50 and Genome fraction % that was closest to the values obtained from the GAGE-B assemblies and supplementary materials.

TABLE 5-1 COMPARISON OF K-VALUES FOR MASURCA 1.8.3 ASSEMBLIES
1) Assemblies downloaded from GAGE-B's webpage
2) Data obtained from GAGE-B's supplementary material

Read type	File type	Assembly	K-value	N50 (kb)	Genome fraction %
MiSeq	Contig	Reproduced	89	61.3	96.9
			99	**76.1**	**97.7**
		GAGE-B Assembly file[1]	N/A	76.1	97.7
		GAGE-B Supplementary material[2]	N/A	76.1	98.3
	Scaffold	Reproduced	89	61.3	96.9
			99	**76.1**	**97.7**
		GAGE-B Assembly file[1]	N/A	76.1	97.7
		GAGE-B Supplementary material[2]	N/A	76.1	98.3
HiSeq	Contig	Reproduced	89	**108.8**	**98.1**
			99	35.3	95.7
		GAGE-B Assembly file[1]	N/A	241.6	98.1
		GAGE-B Supplementary material[2]	N/A	241.6	99.4
	Scaffold	Reproduced	89	**246.8**	**98.2**
			99	46.6	95.8
		GAGE-B Assembly file[1]	N/A	246.5	98.1
		GAGE-B Supplementary material[2]	N/A	246.5	99.3

According to **Error! Reference source not found.** above, the correct k-values for assemblies performed with MaSuRCA appear to be 89 for HiSeq data and 99 for MiSeq data. The comparison of MaSuRCA assemblies will therefore be based upon results from Table B 7 to Table B 10 with these k-values only. For contigs computed with assemblies on HiSeq data, with 1 less local misassembly and 0.005 lower duplication ratio, all the other metrics performed better or equal to the GAGE-B results. The scaffolds computed with assemblies on HiSeq data resulted in a better outcome for the number of scaffolds and duplication ratio. The rest of the metrics were either worse or equal to the values retrieved from the GAGE-B assembly. MaSuRCA assembly on MiSeq data were the only assembly with exactly the same results on all metrics compared to the GAGE-B results.

SOAPDENOVO2 2.04 WITH GAPCLOSER 1.2.12

The results for contigs and scaffolds with 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 numbers of local and global misassemblies are quite low in comparison. Scaffolds on the other hand perform almost as good as the GAGE-B results except for fewer global misassemblies and unaligned contigs, but more local misassemblies.

SPADES 2.3.0

The HiSeq datasets for both contigs and scaffolds in SPAdes 2.3.0 performs better overall compared to GAGE-B results, except for 7 more local misassemblies (contig) and 1 more unaligned contig (contig/scaffold). MiSeq data on the other hand performs equally well as the expected outcome from GAGE-B results.

VELVET 1.2.08

Velvet 1.2.08 performs better when counting the number of contigs, but has in return more scaffolds on assemblies computed with HiSeq data compared to GAGE-B results. The N50 values were the opposite with overall worse values for scaffolds and better values for contigs compared to GAGE-B. The values for NA50 as well as the number of global/local misassemblies were better on all except scaffolds computed with HiSeq data (8 450bp less NA50 and equal amount of # misassemblies compared to GAGE-B results). Contigs produced more local misassemblies while scaffolds produce fewer, but the number of global misassemblies was dependent on data type. Assemblies computed with MiSeq data had fewer # misassemblies, while assemblies computed with HiSeq data had either equal (scaffold) or more # misassemblies (contig). Other than that, the genome fraction, duplication ratio and the number of genes are similar to the GAGE-B results.

5.2.2 COMPARISON OF ASSEMBLIES

The different assemblies in Section 5.2.1 were compared to each other to assess the overall performance of the species *Vibrio cholerae*. Table B 25 and Table B 26 describes the best result for each metric for MiSeq and HiSeq data on both contigs and scaffolds.

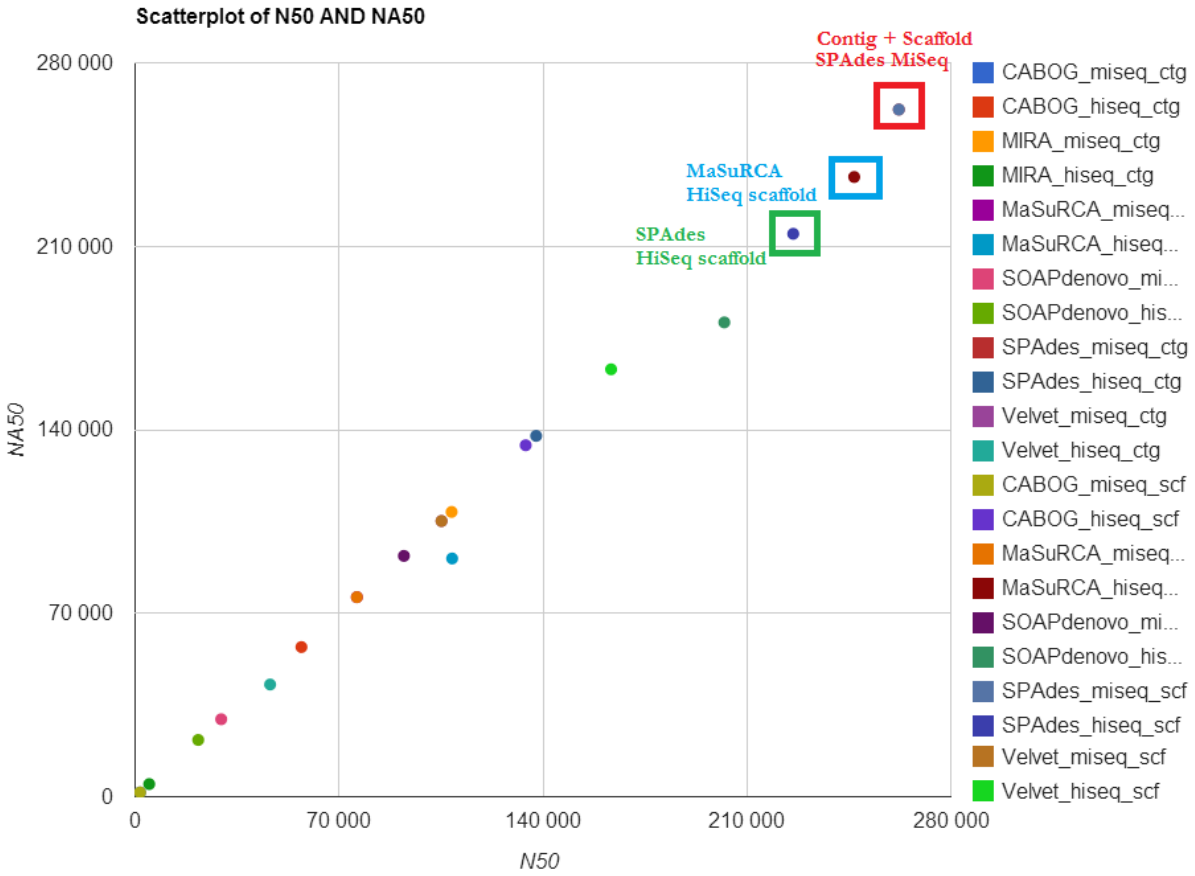


FIGURE 5-1 SCATTERPLOT OF N50 AND NA50 ON REPRODUCED CONTIGS AND SCAFFOLDS

SPAdes have the best result for N50 and NA50 (Figure 5-1), as well as genome fraction and the number of genes (Figure 5-2) overall except for N50 and NA50 for scaffolds computed from HiSeq data where the best results are held by MaSuRCA. MaSuRCA also have the fewest number of contigs while Velvet (133 scaffolds computed on MiSeq data) and SOAPdenovo (77 scaffolds computed on HiSeq data) have the fewest number of scaffolds(Figure S-B 1 and Figure S-B 2).

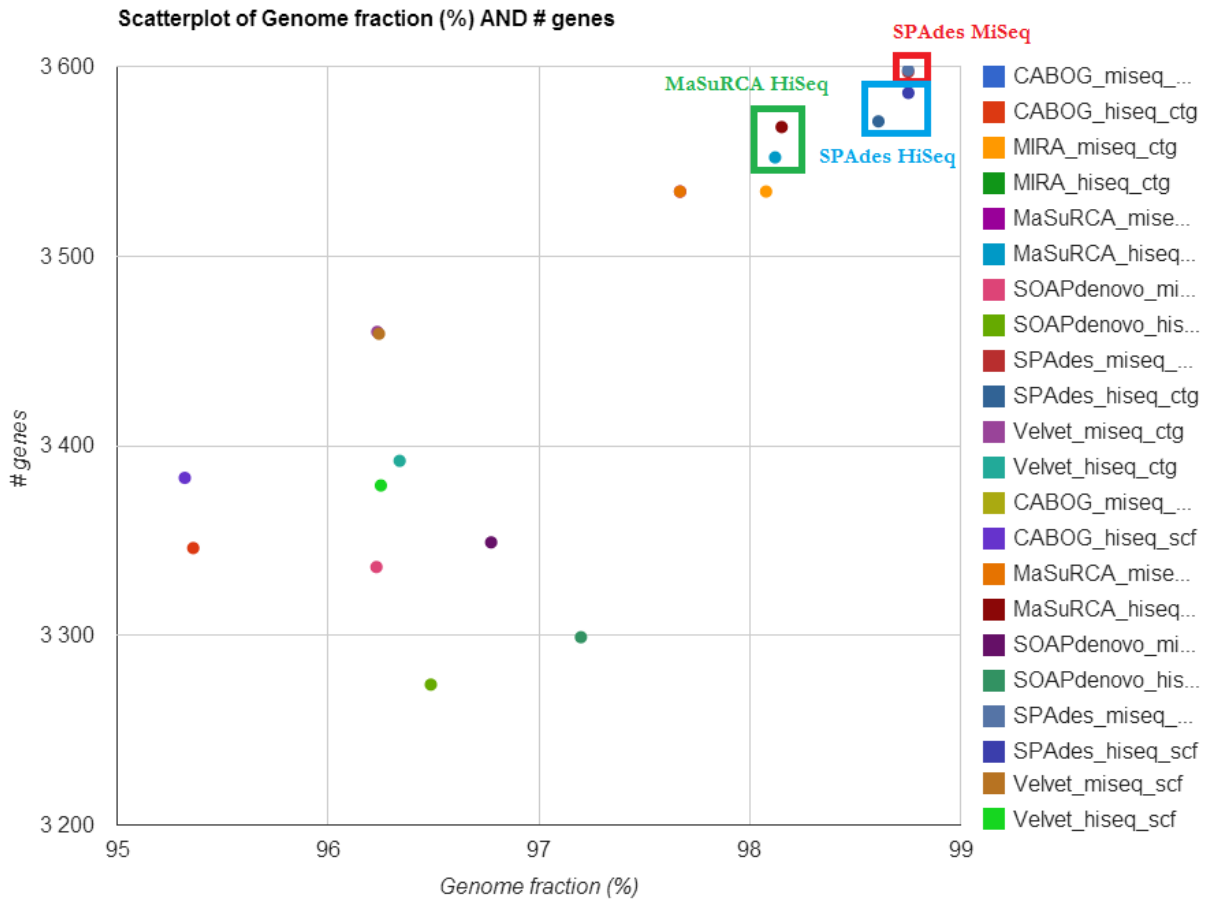


FIGURE 5-2 SCATTERPLOT OF GENOME FRACTION AND THE NUMBER OF GENES OVER THE TENTATIVELY REPRODUCED RESULTS

CABOG MiSeq data and MIRA HiSeq contig have been excluded from the plot to make it easier to view the best results. SPAdes miseq contig is right below SPAdes miseq scaffold, making it a bit difficult to view in this plot. The original version of this plot can be viewed in Figure S-B 3.

Scaffolds computed from HiSeq data have fewer scaffolds than the MiSeq data as Figure 5-3 shows below. SOAPdenovo have the lowest duplication ratio in 3 out of 4 cases with scaffolds computed from MiSeq data being the exception held by SPAdes.

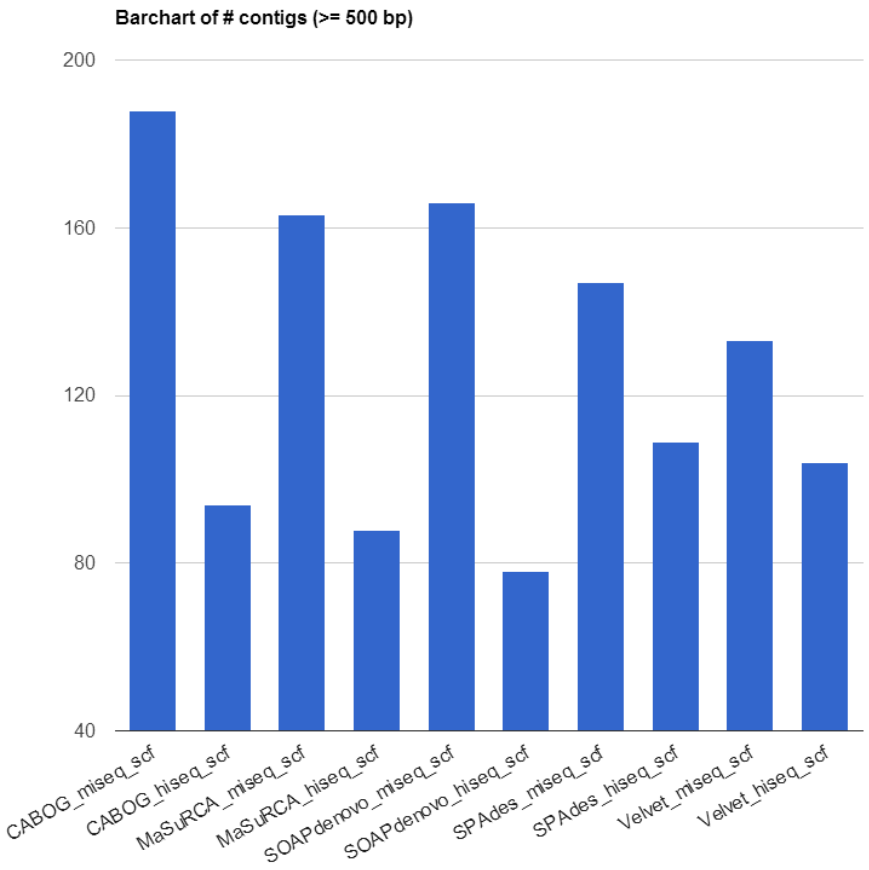


FIGURE 5-3 SCAFFOLDS FROM REPRODUCED GAGE-B RESULTS

5.3 REUSEABILITY OF GAGE-B DATA

This section describes results retrieved on the same dataset as in Section 5.2, but with newer versions of each assembler if available. Assemblies obtained with ABySS and SGA are still excluded from the comparison based on the unsuccessfully runs as mentioned in Section 5.2.1. Other assemblers excluded from this comparison are newer versions of MIRA (because of installation problems) and SOAPdenovo2 (because of the absence of new releases), leaving newer versions of CABOG, MaSuRCA, SPAdes and Velvet to be compared.

5.3.1 ASSEMBLER SPECIFIC COMPARISON – NEW ASSEMBLER VERSIONS

This section contains assembler specific comparison of new assembler version with the old version used in the GAGE-B paper. Each assembler is compared in separate sections.

CABOG 8.1

Assemblies computed on HiSeq data with CABOG 8.1 performs worse than its previous version (7.0) except for fewer misassemblies (contigs/scaffolds), equal amount of local misassemblies (contigs) and unaligned contigs (scaffolds). The duplication ratio is also lower by 0.004 for contigs. Assemblies computed on MiSeq data have more contigs/scaffolds and number of global/local misassemblies, but are still performing better on all other metrics compared to the original version 7.0.

MASURCA 2.1.0

Assemblies computed with MaSuRCA 2.1.0 performs worse on most metrics for both MiSeq and HiSeq data. The exception is higher genome fraction and equal amount of global misassemblies for assemblies computed on HiSeq data, and fewer global misassemblies and unaligned contigs (contigs/scaffolds) for assemblies computed on MiSeq data.

SPADES 2.5.0

SPAdes 2.5.0 assemblies on HiSeq data returned better results for the number of contigs/scaffolds, N50, NA50, local misassemblies, and duplication ratio. While the number of unaligned contigs/scaffolds was the same for both versions, the rest of the metrics returned better result for previous version 2.3.0. SPAdes 2.5.0 assemblies on MiSeq data returned lower values for all metrics, (resulting in for instance better # misassemblies, but worse N50) except for 37 more scaffolds and 24 more unaligned contigs.

VELVET 1.2.10

Assemblies computed by Velvet 1.2.10 gave the exact same results as assemblies computed by Velvet 1.2.08 for both HiSeq and MiSeq data.

5.3.2 COMPARISON OF ASSEMBLIES – NEW ASSEMBLER VERSIONS

The different assemblies in Section 5.3.1 were compared to each other to assess the overall performance of the species *Vibrio cholerae* on the new assembler versions. Table B 27 and

Table B 28 describes the best result for each metric for MiSeq and HiSeq data on both contigs and scaffolds.

Velvet has the fewest number of contigs and scaffold for MiSeq data while SPAdes have the fewest number of contigs and scaffolds for HiSeq data. CABOG assemblies have the fewest number of unaligned contigs/scaffolds overall, but share the position with MaSuRCA for MiSeq data and Velvet for HiSeq contigs. SPAdes have the highest number of unaligned contigs/scaffolds on MiSeq assemblies, but can boast with the best results for the remaining metrics for MiSeq data. SPAdes also has the best results for N50, NA50 and the number of global/local misassemblies for assemblies computed on HiSeq data, as well as second place for the remaining metrics after CABOG/MaSuRCA.

CHAPTER 6 DISCUSSION

This chapter will cover challenges related to the implementation of the Galaxy tools, the reproduction of the GAGE-B results, reusability of data with new assembler version, as well as possible solution to some problems (Section 6.1). It will also interpret all results (Section 6.2) described in Chapter 5 and give an analysis of the Galaxy tools developed for assembly evaluation (Section 6.3). The chapter is wrapped up with a section about further work (Section 6.4) followed by an overall conclusion (Section 6.5).

6.1 CHALLENGES ENCOUNTERED DURING THE ASSEMBLY RUNS AND IMPLEMENTATION

There have been quite a number of obstacles throughout the thesis. Everything from errors while installing new software and unsuccessful assemblies to the struggle with server issues that is way beyond one's access level. Below is an attempt to summarize the most time consuming and perhaps remarkable events encountered.

6.1.1 GALAXY FRAMEWORK

The initial Framework was easy to retrieve with a simple git command since it's required dependencies was preinstalled on the original server used. Even though at times the Galaxy's "user manual" was lacking, local expertise from the University was very helpful and highly appreciated. Unfortunately, the initial problems started with the server updates. Suddenly the Galaxy instance was not working anymore and the ball started rolling from there. The Galaxy instance had to be moved to another server where not all dependencies were preinstalled, making it quite a time and resource consuming task to get the instance up and running. Later on the QUAST runs did not finish and another round with debugging was started. The solution to this problem was that, for some unknown reason, QUAST could not run with --static which had to be removed from its Makefile. Finally the new Galaxy tools for assembly evaluation ran properly, but there was one last problem to be fixed. One Python path in the server's login script made it impossible to upload new files to Galaxy, because it mixed the Galaxy code with another version used by Hyperbrowser developers using the same server. The problem was discovered fast and the solution was simple giving a properly running Galaxy instance as soon as the Python path was excluded from the login script.

6.1.2 REPRODUCING GAGE-B RESULTS

Reproducing GAGE-B results was undoubtedly the most time consuming part of this thesis. Not only were there a lot of files, assemblers and assemblies to keep track on, but almost nothing went smoothly either. For starters, installing each assembler was quite time consuming because usually an installation will run until it encounter an error and exit, sometimes with an incomprehensible error report other times with a simple missing dependency which had to be installed separately. Knowing which dependency versions each assembler need would probably have reduced the installation time significantly. With this being said, most assemblers have a good documentation, but it's hard to tell which version of the dependencies the GAGE-B authors used, and if the versions would have made a difference on the outcome. For instance with ABySS

where boost, sparsehash and open MPI as well as a C++ compiler that supports OpenMPI is required, there's a lot that need to work together to produce the assemblies. Another thing with this assembler was the installation option *--enable-maxk*. Compiling ABySS the first time without prior knowledge resulted in unsuccessful runs for MiSeq data on *Vibrio cholerae* since it was supposed to be run with k=65 and the default maximum k-value is 64. Maybe a text file with installation options other than the defaults could have spared some compilation and installation time. This is usually not considered to be a high priority task and even for this thesis making a file with installation commands/options was unfortunately not done. It was difficult to keep track of the installation options and the versions of the server modules since some assemblers had to be re-installed several times before they satisfied the **"dependency hell"** or adjusted to the various server updates.

Another problem encountered upon reproduction of the GAGE-B results was the recipe created by the authors. Examples of some of the errors encountered are described in Table A 4. Not only did the authors save the recipe as a pdf file making copying some commands correctly almost impossible, but the recipe itself contained misspellings resulting in faulty runs. Some runs also had missing options to run properly like the SGA assembly with HiSeq data missing the --phred64 option. Errors like those mentioned above made the reproduction significantly harder, while debugging took quite an amount of time which probably could have been avoided if the recipe had been saved as a text file and been proofread before publication.

Even though the reproduction was filled with complications that made it impossible to reproduce all results in the thesis' time frame, the focus on only one species was not enough to compute the exact same results as in the paper or the assembly files provided by the authors. It's difficult to point out exactly what caused the differences between the reproduced assemblies and the GAGE-B results, but some possible explanation to the problems can be assumed. These explanations are elaborated upon in the next four sub-sections.

ASSEMBLER ENVIRONMENT

As mentioned earlier, assemblers have many dependencies, and it can be reasonable to assume that maybe some dependency versions affect the outcome of an assembly. The ideal solution for this issue can be to either create a Makefile, or a virtual machine to make it possible to reproduce results in the same environment as it was computed on originally. This is of course a time and resource consuming task, but would most likely be highly appreciated by anyone trying to reproduce a paper's results.

READ TYPE DIFFERENCES

It can be possible that a given set of reads are not the same as the ones used in the paper. This could explain differences in the original assemblies compared to the reproduced ones as the input is not the same. Perhaps the reads are different or maybe raw reads have been used were clean/trimmed ones were supposed to be used or vice versa. It should be unnecessary for someone reproducing the results to double check if the information about the read types is correct or not. It would be justifiable to assume that the papers authors made sure that the reads uploaded are in fact the same as the ones used as input for their assemblies and that the information provided about raw or clean/trimmed reads are correct.

DIFFERENT ASSEMBLIES

It is plausible to assume that the assemblies that can be downloaded are not the same as those used to create the supplementary tables in the GAGE-B paper since they differ from each other

as described in Table B 23 and Table B 24. Perhaps several assemblies were computed and the reproduced results match an assembly that has been excluded from the paper and the download files. Once again, making sure that the assemblies uploaded are in fact the ones used in the paper can reduce some differences in the reproduced results.

TOOL OPTIONS

Since inconsistency between supplementary material and assembly files are observed, and there were discovered problems with the GAGE-B recipe, it might be reasonable to assume that some assembler related information is missing or falsely added. One possible explanation may be that the GAGE-B authors used additional options when computing the assemblies that were left out from the recipe. Another possible explanation could be that some options written in the recipe were not used in their assemblies at all. It can also be possible that QUAST generated different outputs based on options used by the GAGE-B authors, but it's impossible to tell since no options for QUAST were mentioned in the paper.

STOCHASTICITY

Last, but not least, maybe the most plausible explanation to different results is stochasticity of some assemblers. This means that some assemblers may return different outputs even though the assemblies are computed with the same options each time. This is unfortunately difficult to deal with as predicting the exact results is near to impossible. The solution to this issue can be to compute several assemblies and calculate an average to confirm if the results point to the same conclusions as in the original assembly or not. Researchers that compute assemblies with stochastic assemblers (whether or not an average of multiple assemblies are calculated) should mention this is their paper if they are aware of this.

6.2 INTERPRETING THE RESULTS

In this section we will interpret the results described in Chapter 5. Some plots on the discussed results can be viewed in the supplementary files *Supplementary_Figures_A* (Figure S-A) and *Supplementary_Figures_B* (Figure S-B) which can be accessed from https://github.com/subway/masterthesis/blob/master/Supplementary%20Material. The interpretation will cover assessment on

1. all species from **Error! Reference source not found.**
2. reproduced assemblies on *Vibrio cholerae*
3. reproduced assemblies on *Vibrio cholerae* with new assembler versions

6.2.1 INCONSISTENT GAGE-B RESULTS

In this section we will try to assess the results from the GAGE-B assemblies as described in Section 5.1 which clearly states the many differences observed between the paper's supplementary materials and assembly files on *Vibrio cholerae*. This section will also take advantage of the new Galaxy tools for assembly evaluation and try to evaluate if the assembly files over all species result in the same conclusion as in the GAGE-B paper or not.

> *"In the MiSeq assembly of B.cereus (Supplementary Table S1), MaSuRCA and SOAPdenovo generated contigs with the highest N50 and corrected N50 values. […] SOAPdenovo generated*

the largest scaffolds, but again had more local errors than most other assemblies. All assemblies covered >99% of the genome by both contigs and scaffolds." [2]

According to the assembly files available on GAGE-B's website, MaSuRCA and SOAPdenovo did generate contigs with the highest N50 and corrected N50 (NA50) as observed in Figure S-A 1, and SOAPdenovo did have more local errors (misassemblies) than most assemblies. The exception is Velvet which had 222 more local errors (misassemblies) as Figure S-A 2 shows. Not all assemblies covered >99% of the genome by both contigs and scaffolds though. Only the red bars (CABOG contig and scaffold, MIRA contig and MaSuRCA contig and scaffold) in Figure 6-1 indicate a genome fraction over 99%, the rest covers between 97.572-98.997% of the genome. This might indicate that different assemblies or reference genomes were used in the paper and the uploaded assembly files.

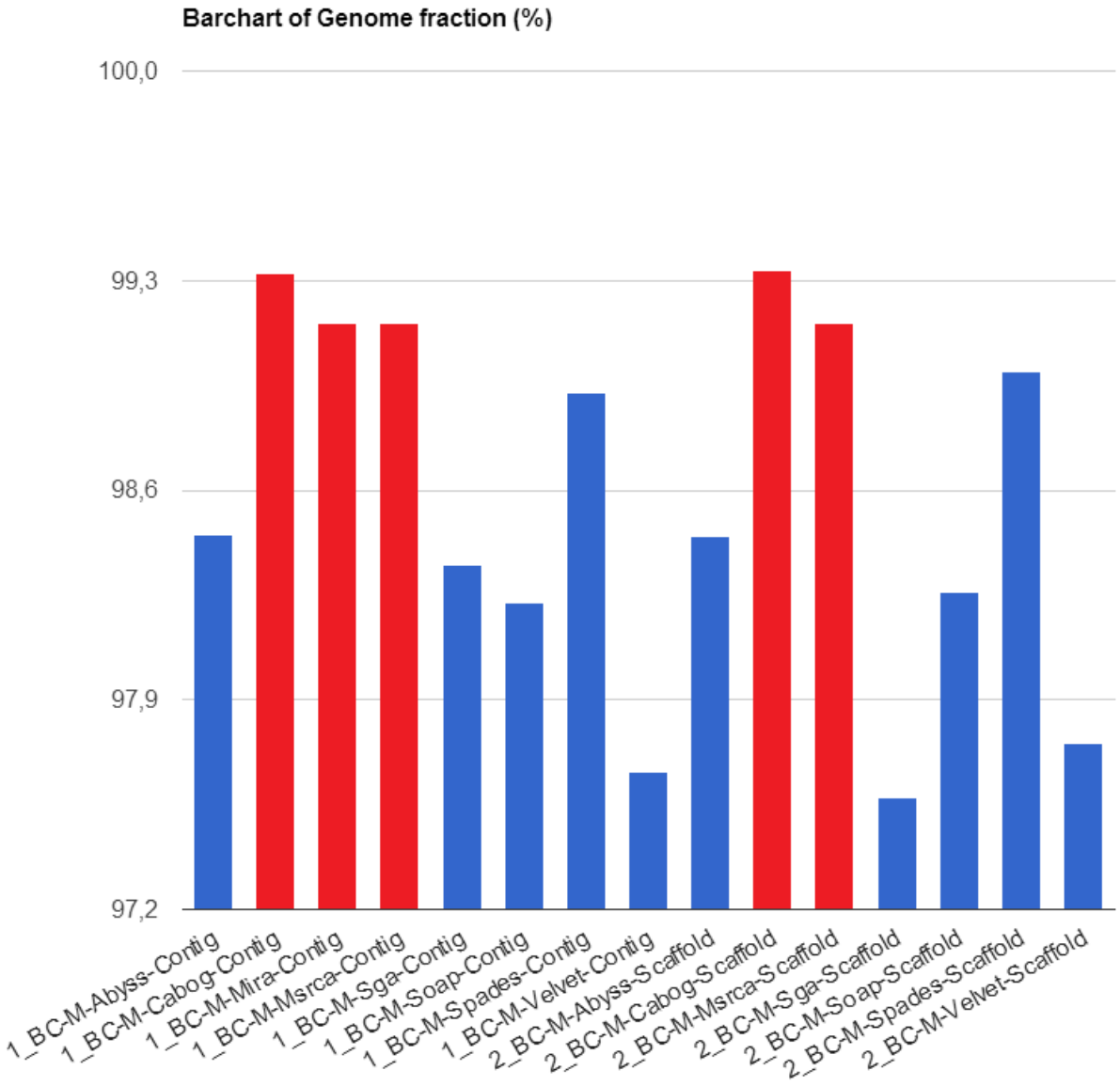


FIGURE 6-1 GENOME FRACTION OF MISEQ ASSEMBLY ON *BACILLUS CEREUS*

> *"For the HiSeq assemblies of R.sphaeroides (Supplementary Table S2), MaSuRCA had the highest contig N50 values at 176.8kb, followed by SPAdes at 83.5kb. All of the other assemblies were far more fragmented, with N50 sizes ranging from 10.1 to 17.7kb. Thus, for this genome, the choice of assembler seems to have a large impact on the quality of the resulting assembly. Looking at the MiSeq data for the same genome (Supplementary Table S3), the results are similar: MaSuRCA and SPAdes produced large contigs, and most other assemblers had N50 values four to five times smaller. The results for scaffolds showed the same relative performance." [2]*

For the HiSeq assemblies of *Rhodobacter sphaeroides* (Figure S-A 3) MaSuRCA had the highest N50 values at 176.8kb followed by SPAdes at 74.5kb, which is 9kb lower than the value written in the paper. The rest of the N50 sizes ranged from 11.1 to 18.7kb which, once again, does not correspond with the values written in the paper. Despite minor differences, it turns out that it is correct to assume that the choice of assembler seems to have a large impact on the quality of the resulting assembly. The results for the MiSeq assemblies on the same genome (Figure S-A 4) are similar, assuming that "large contigs" means high N50 values. MaSuRCA had the highest N50 value at 142.7kb followed by SPAdes at 118.1kb whereas most other assemblers had N50 values 3-6 times smaller (total sizes ranging from 9.2-41.8kb). The results for scaffolds showed the same relative performance in the assembly files too.

> *"For the M.abscessus assemblies (Supplementary Tables S4 and S5), MaSuRCA had the largest contig N50 size at 246.9kb. The other assemblers—MIRA, SOAPdenovo and SPAdes—had N50 sizes of ~150kb for the HiSeq data. Noteworthy here is that although most assemblers performed worse with the MiSeq data, SPAdes performed considerably better, with an N50 contig size of 215.4kb." [2]*

The assembly files provided by the GAGE-B authors also resulted in largest N50 on HiSeq assemblies (blue bars in Figure 6-2) computed by MaSuRCA (246.8kb) while HiSeq assemblies for MIRA, SOAPdenovo and SPAdes came in second with a N50 of roughly 150kb. All assemblers perform worse with the MiSeq data (green bars in Figure 6-2), except for SPAdes (220.1kb better, and more than what the paper presents) as expected according to the paper.

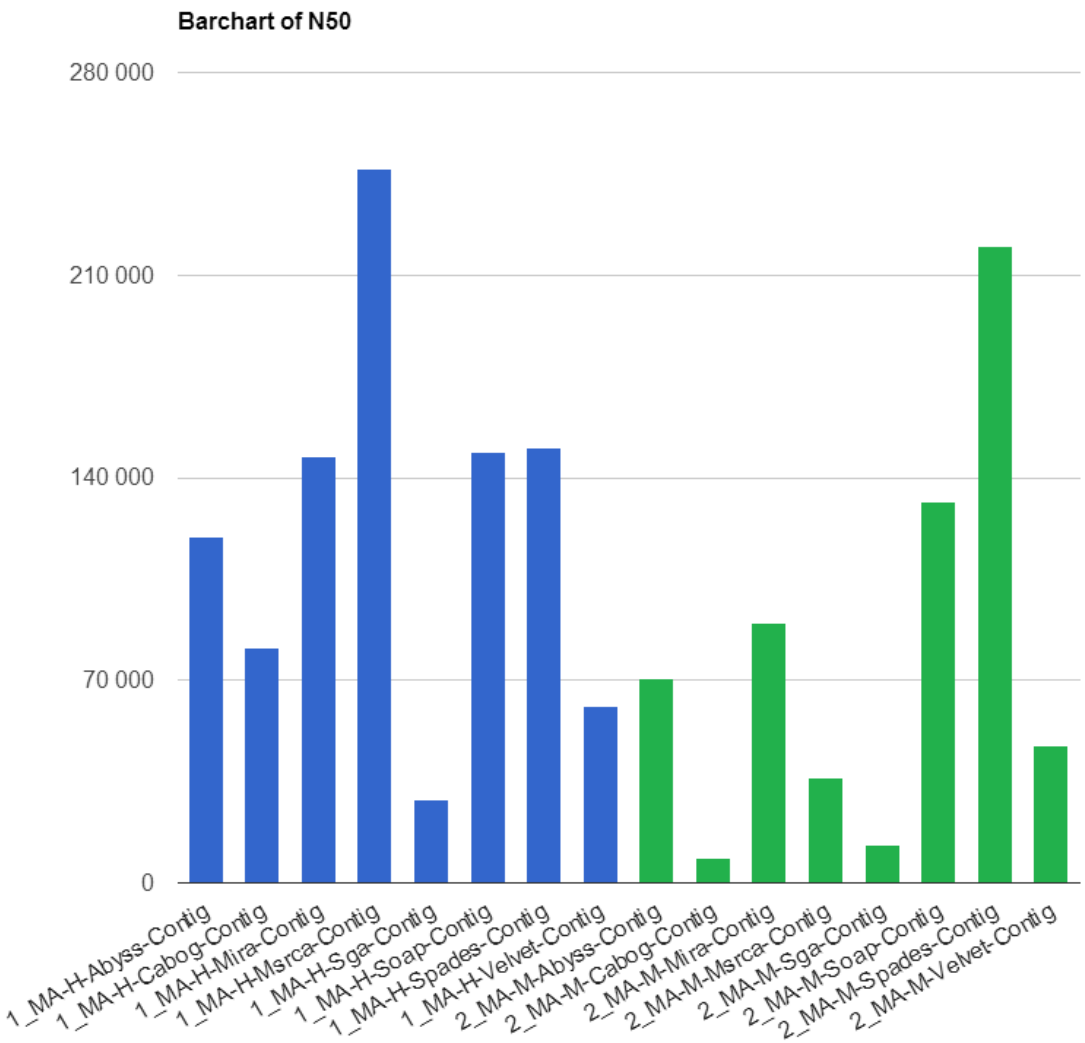


FIGURE 6-2 N50 STATISTICS FOR CONTIGS FROM HISEQ (1-BLUE) AND MISEQ (2-GREEN) ASSEMBLIES ON *MYCOBACTERIUM ABSCESSUS*

> *"The largest contigs for V.cholerae (Supplementary Tables S6 and S7) were produced by SPAdes from 250bp reads. MaSuRCA produced contigs that were nearly as large from the 100bp data. Both assemblers produced contig N50 sizes >225kb, whereas most other assemblers were <100kb. SOAPdenovo and MIRA produced contig N50 sizes close to 100kb, but with a larger number of errors than most other methods." [2]*

The largest contigs for *Vibrio cholerae* on the uploaded assembly files (Figure S-A 5 for HiSeq data and Figure S-A 6 for MiSeq data) were produced, just as in the paper, by SPAdes and MaSuRCA with N50 sizes larger than 225kb. Most other assemblers N50 sizes were less than 100kb, except for SPAdes (HiSeq) and MIRA (MiSeq), which also had largest number of errors (Figure S-A 7 and Figure S-A 8) as stated in the GAGE-B paper.

> *"The A.hydrophila assembly (Supplementary Table S8) is particularly worth noting because of the remarkably large N50 size that MaSuRCA produced, 828.6kb. The assembly contained only 32 contigs for this 4.75Mb genome, the smallest number for any of our experiments. Most of the other assemblers also produced large contigs for this genome, indicating that it was the 'easiest' to assemble of all the GAGE-B datasets" [2]*

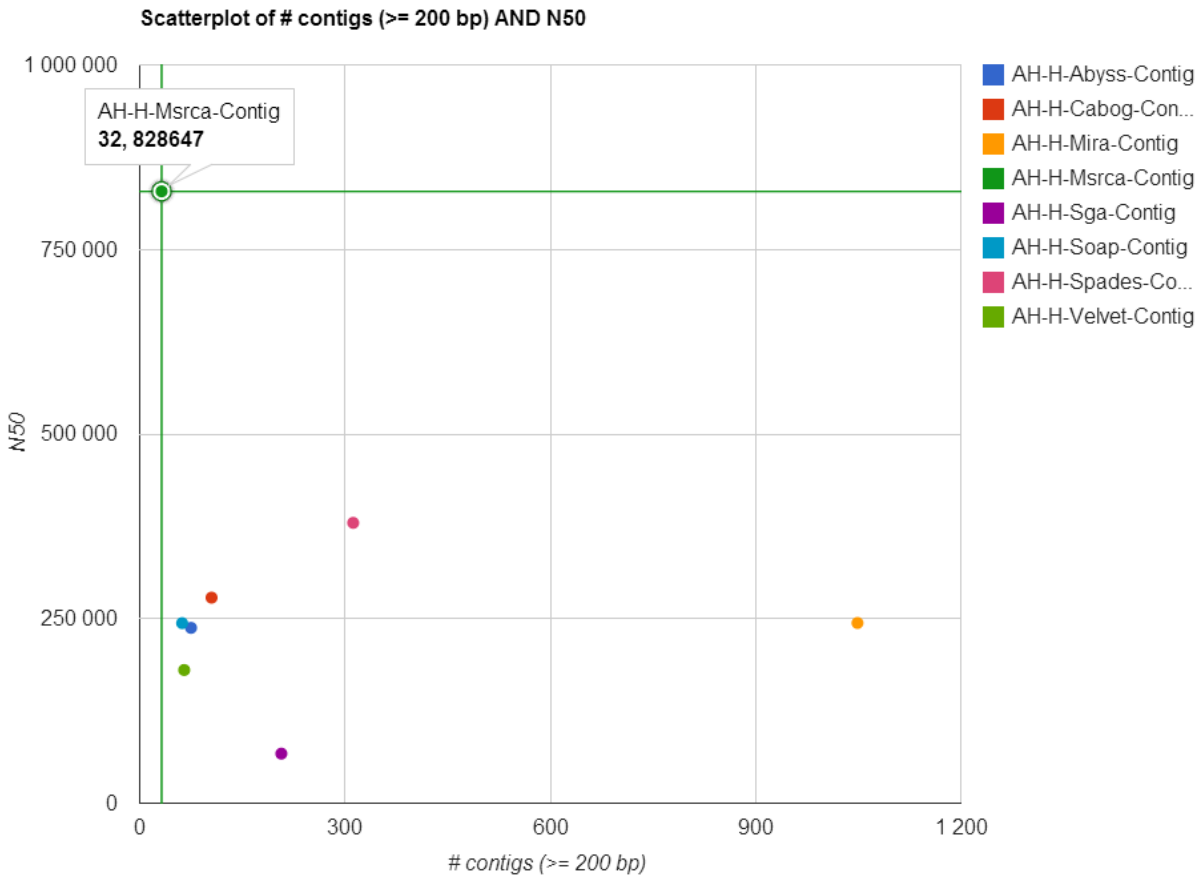


FIGURE 6-3 SCATTERPLOT OF THE NUMBER OF CONTIGS AND N50 ON CONTIGS FROM HISEQ ASSEMBLIES ON *AEROMONAS HYDROPHILA*

Figure 6-3 above confirms that the uploaded assembly files on *Aeromonas hydrophila* also have a remarkably high N50 and lowest number of contigs produced by MaSuRCA. Most assemblers also produced high N50 values (Figure S-A 9), thus correlating with the papers observations and indicating that this genome might have been "easier" to assemble than others.

> *"At the other end of the spectrum, the B.cereus VD118 strain, which was only available in 100bp reads, presented the greatest difficulties for all of the assemblers. The best contig N50 sizes were just above and below 10kb, and no assembly had fewer than 164 contigs. The other B.cereus strain yielded better assemblies, but because this was based on 250bp reads, it is hard to ascertain the precise reason for the differences." [2]*

The N50 (Figure S-A 10) values for the GAGE-B's HiSeq assembly files on the *Bacillus cereus VD118* strain does not correspond with the papers description at all. The contig N50 sizes (Figure 6-4) ranged from 23.4 to 97.2kb which is way above the "just above and below 10kb" stated in the paper. Other than that no assembly had fewer than 164 contigs, and the other *Bacillus cereus* strain (ATCC 10987) did indeed yield better assemblies compared to the same metrics (Figure S-A 11).

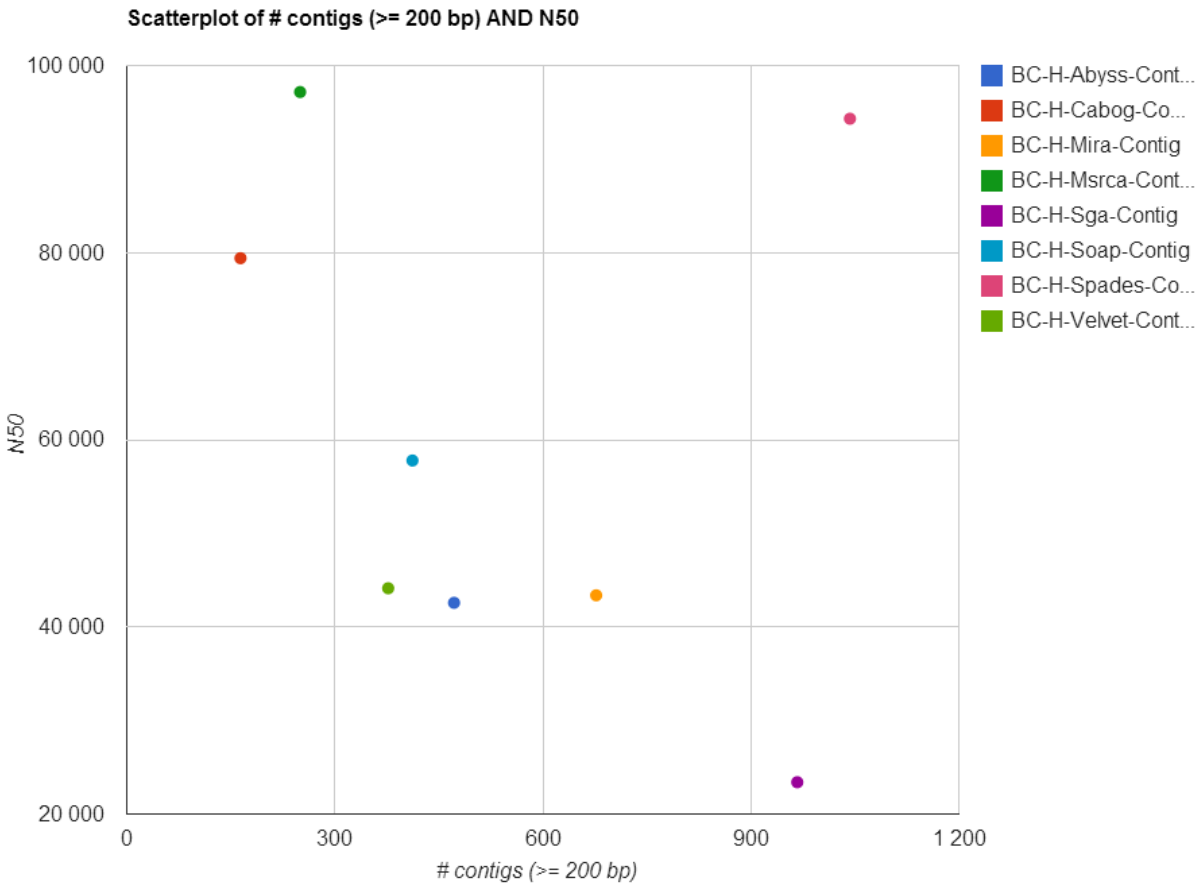


FIGURE 6-4 SCATTERPLOT OF THE NUMBER OF CONTIGS AND N50 ON CONTIGS FROM HISEQ ASSEMBLIES ON *BACILLUS CEREUS VD118*

> *"Although no assembler won on all the various metrics, the MaSuRCA assembler had the largest contig sizes, measured by either N50 or corrected N50 values, for 10 of the 12 experiments. The SPAdes assembler, a relatively recent entry into the next-generation assembly field, came in first or essentially tied for first for 4 of the 12 genomes."* [2]

According to the assembly files, MaSuRCA does indeed have the largest contig sizes, measured by either N50 or corrected N50 values (NA50) for 10 of the 12 experiments. This can be viewed in Figure 6-5 for HiSeq data and Figure 6-6 for MiSeq data, or in full size in Figure S-A 1, Figure S-A 5, Figure S-A 6, Figure S-A 9, Figure S-A 10, and from Figure S-A 12 to Figure S-A 18. SPAdes did, as mentioned in the paper, come in first or essentially tied for 4 of the 12 genomes.

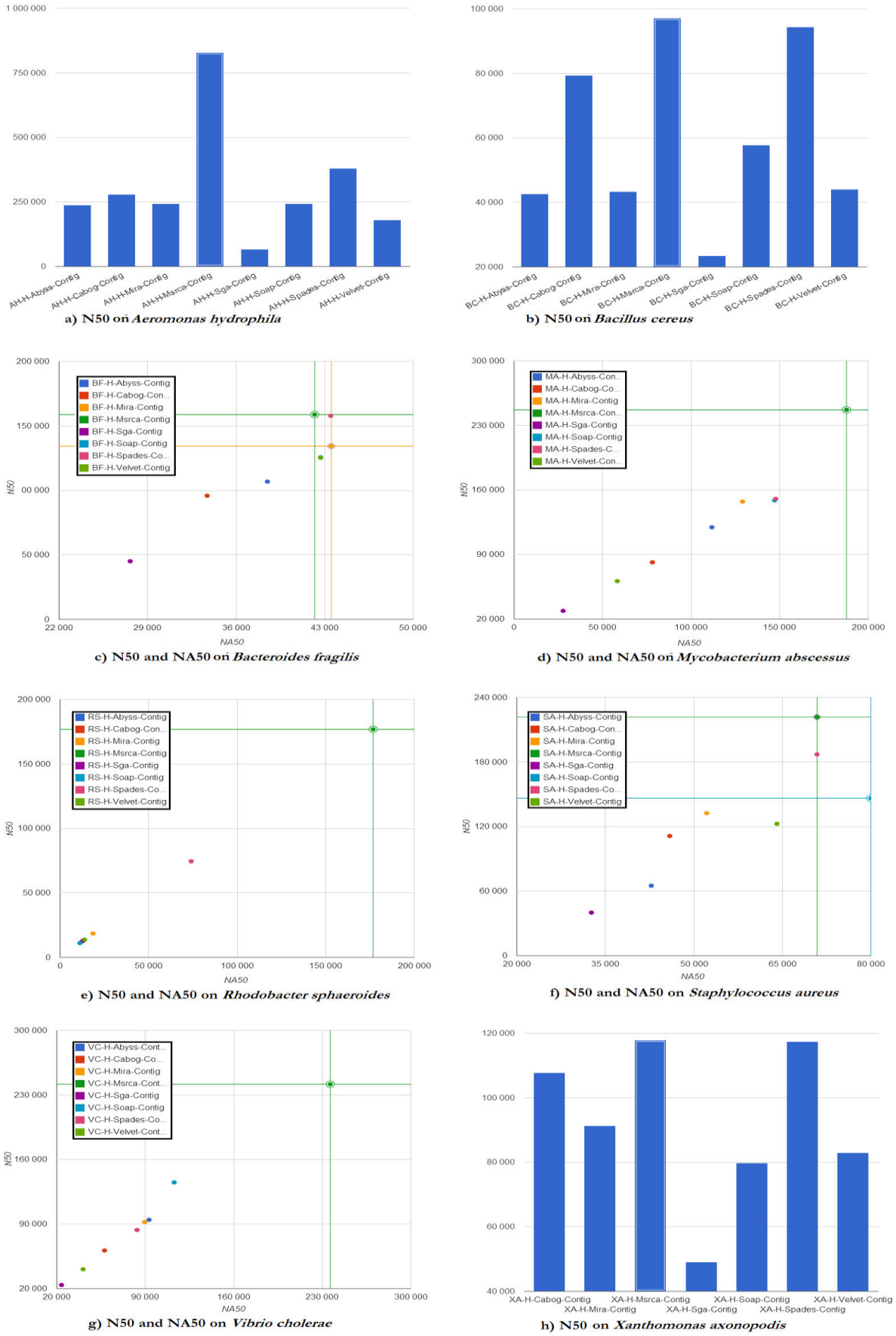
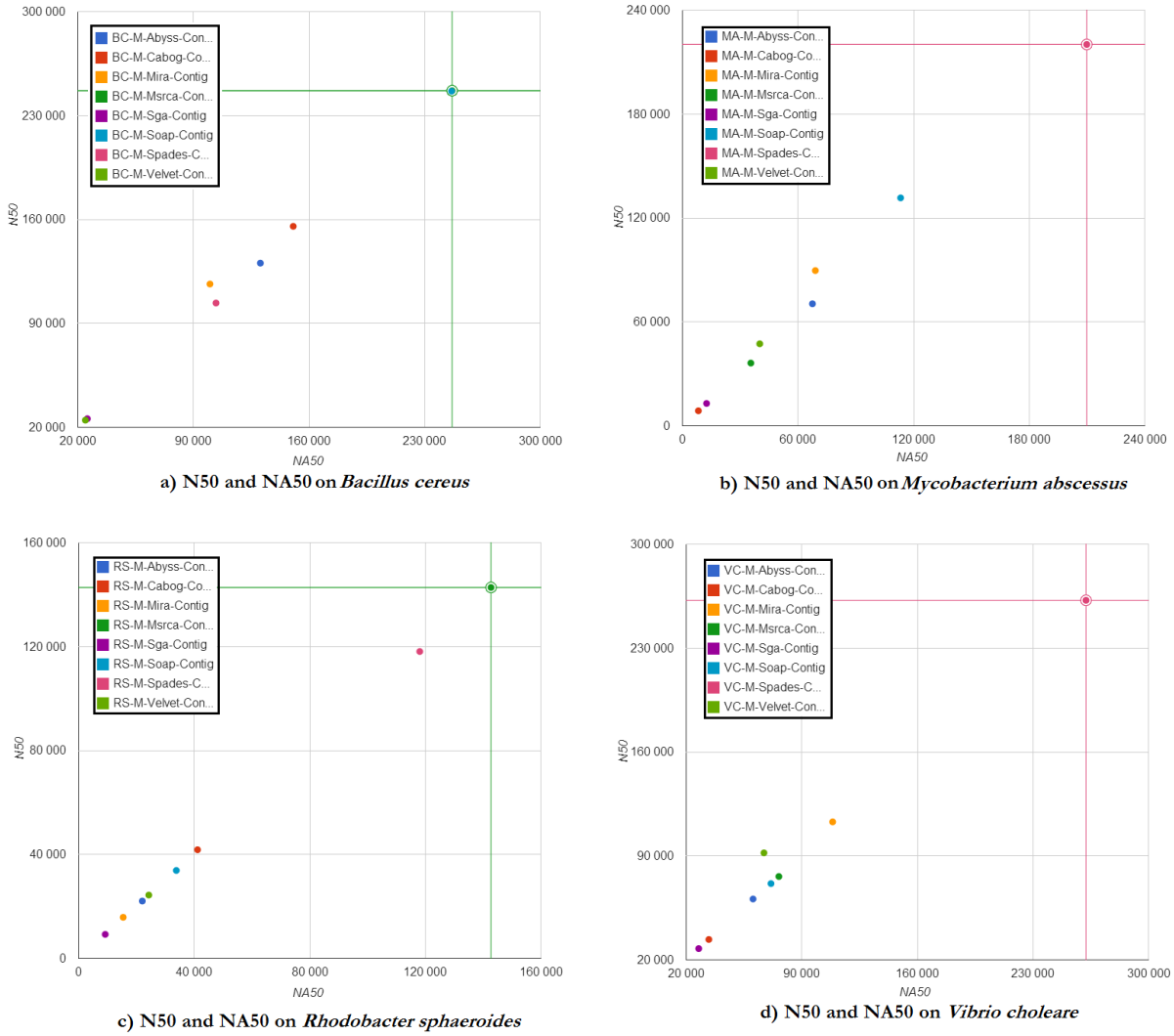


FIGURE 6-5 PLOTS ON N50 AND NA50 (WERE POSSIBLE) ON 8 SPECIES' HISEQ DATA



a) N50 and NA50 on *Bacillus cereus*

b) N50 and NA50 on *Mycobacterium abscessus*

c) N50 and NA50 on *Rhodobacter sphaeroides*

d) N50 and NA50 on *Vibrio choleare*

FIGURE 6-6 SCATTERPLOT ON N50 AND NA50 ON FOUR DIFFERENT SPECIES' MISEQ DATA

> *When considering the number of errors, including local errors, ABySS and SGA consistently produced assemblies with the fewest errors. These assemblers also tended to produce smaller contigs than most of the others, suggesting that they use a conservative assembly strategy that trades off contig size for accuracy.* [2]

Unlike what's written in the paper, ABySS and SGA produced assemblies with the fewest errors on all species (Figure S-A 19 to Figure S-A 29) except *Xanthomonas axonopodis* (Figure 6-7) where both assembler have the most errors. The contig size measured by N50 on the other hand is consistent with the paper's suggestion that these assemblers use a conservative assembly strategy that trades off contig size for accuracy.

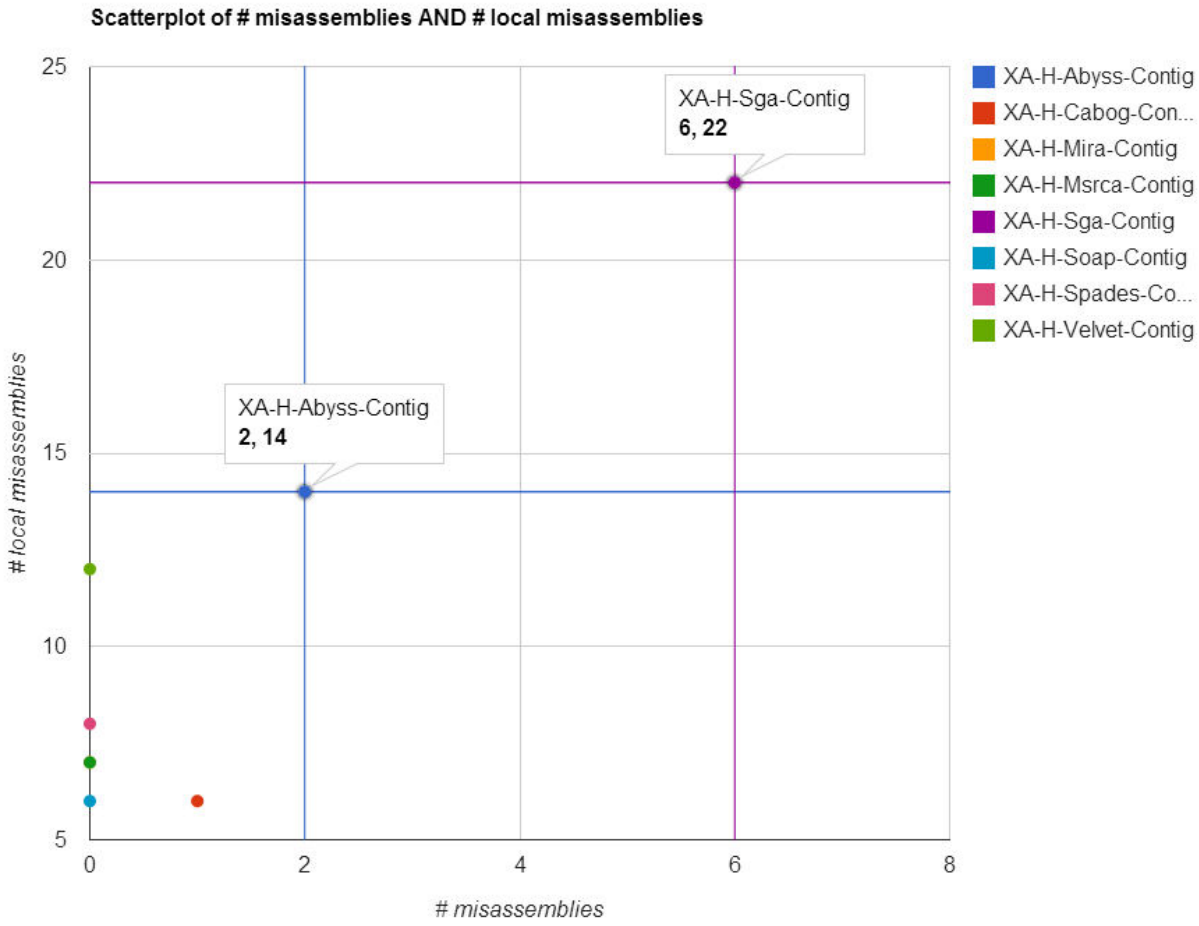


FIGURE 6-7 SCATTERPLOT OF # MISASSEMBLIES AND # LOCAL MISASSEMBLIES ON CONTIGS FROM HISEQ ASSEMBLIES ON *XANTHOMONAS AXONOPODIS*

> *Overall, MaSuRCA and SPAdes produced the best assemblies across these 12 bacterial organisms.*[2]

Despite the slightly different results for some species, the conclusion is still the same with MaSuRCA and SPAdes producing the best assemblies across these 12 bacterial organisms.

6.2.2 REPRODUCING GAGE-B RESULTS

In this section we will assess the results from the reproduction of GAGE-B assemblies on *Vibrio cholerae* as described in Section 5.2. Conclusions in this section are supported by the supplementary files *Supplementary_Tables* and *Supplementary_Figures_B* available on the Github site[6].

Starting with the N50 and NA50 (Figure 5-1), the assembler with the highest values for both HiSeq and MiSeq data is SPAdes followed by MaSuRCA on HiSeq scaffolds only. SPAdes assemblies (HiSeq and MiSeq) also covered the highest fraction of the genome (>98.6%), with a strong second by MaSuRCA (98.1% for HiSeq data). The number of genes replacing the GAGE-B metric "# proteins" had, once again, SPAdes as the assembler with the highest values with MaSuRCA as a close second (Figure 5-2).

[6] https://github.com/subway/masterthesis/tree/master/Supplementary%20Material

When considering the number of global and local misassemblies, SPAdes (HiSeq) and CABOG (MiSeq) rank at first place for the lowest number on scaffold while SOAPdenovo outrank everyone on contigs as well as having the lowest number of unaligned contigs/scaffolds on MiSeq data. Unfortunately for the latter one, that is about as good as it gets. An interesting statistic is observed in the correlation between the number of contigs and unaligned contigs for assemblies computed on SPAdes suggesting, as in the GAGE-B paper, that it might sometimes compute many small contigs without a proper alignment to the reference genome. This, accompanied by everything else observed above, suggestion that the overall best assembler is SPAdes with MaSuRCA close behind, which is almost the same as the conclusion drawn by the GAGE-B authors where both assemblers score equally.

6.2.3 REUSABILITY OF GAGE-B RESULTS

This section will evaluate the development of assemblers by looking at the capabilities of the various new assembler versions used in this thesis. Is it realistic to predict the same conclusions as in Section 6.2.2 or have some assemblers had a performance boost?

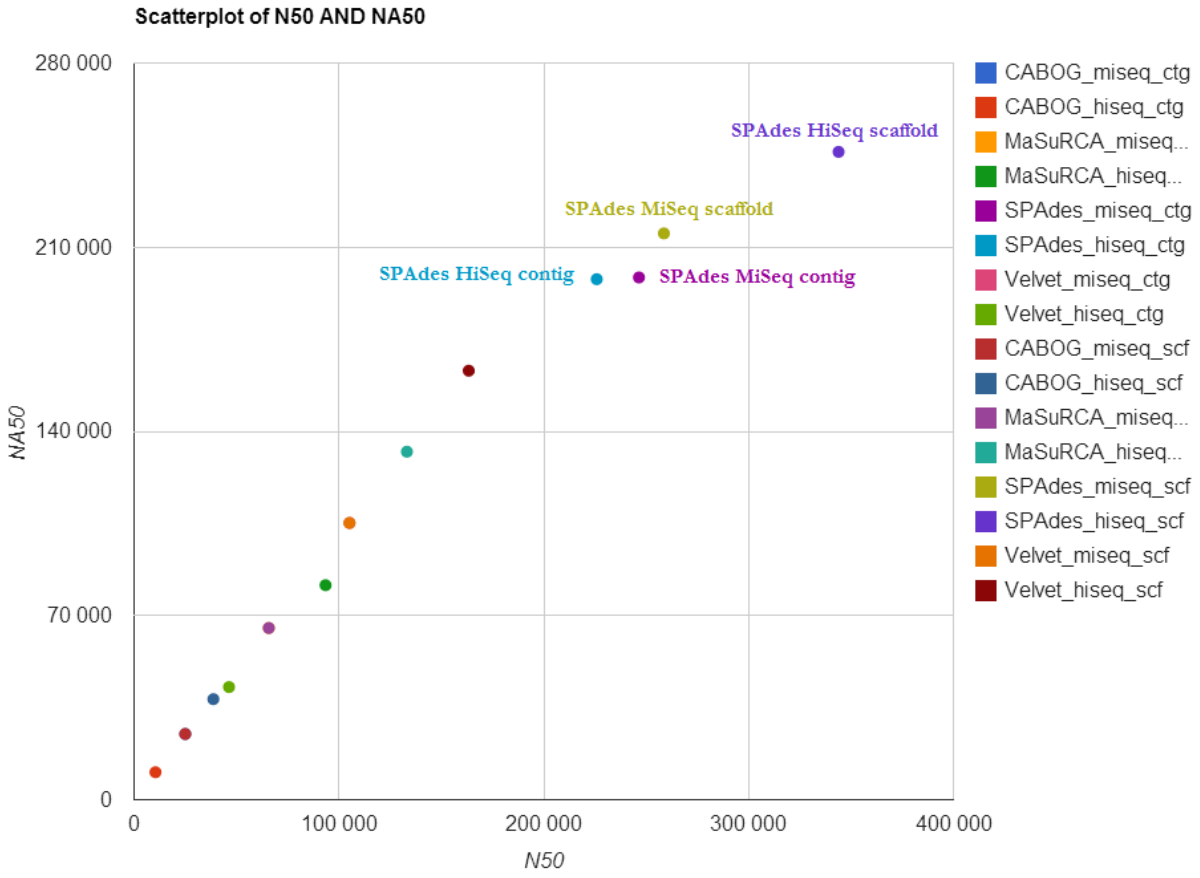


FIGURE 6-8 SCATTERPLOT OF N50 AND NA50 ON REPRODUCED RESULTS WITH NEW ASSEMBLER VERSIONS

Once again SPAdes produced the best results for N50 and NA50 overall even though the new versions gave worse N50 (3.5-15.7 kb less) and NA50 (46.9-63.7 kb less) on assemblies with MiSeq data (Figure 6-8). Assemblies with HiSeq data had a remarkable boost of 88.2-118.1 kb for N50 and 31.5-60.2 kb for NA50. SPAdes had the highest genome fraction, number of genes and duplication ratio on assemblies with MiSeq data even though the first two metrics were better on previous versions. With a slight improvement of genome fraction from previous version, but

with a reduced number of genes, MaSuRCA was able to give the best results for these metrics on assemblies with HiSeq data.

Except for 5 more global misassemblies on assemblies with HiSeq data, SPAdes had the fewest number of global/local misassemblies with a reduction of 1 misassembly and 3-8 local misassemblies. MaSuRCA managed to reduce one unaligned contig with the new version on MiSeq data resulting in the same amount of unaligned contigs as CABOG assemblies, while an increased unaligned contig for CABOG assembly on HiSeq data resulted in the same amount of unaligned contigs as Velvet assembly on HiSeq contigs. Nonetheless, the fewest unaligned contigs are still computed using CABOG with a massive reduction of 96 unaligned contigs on assemblies with MiSeq data.

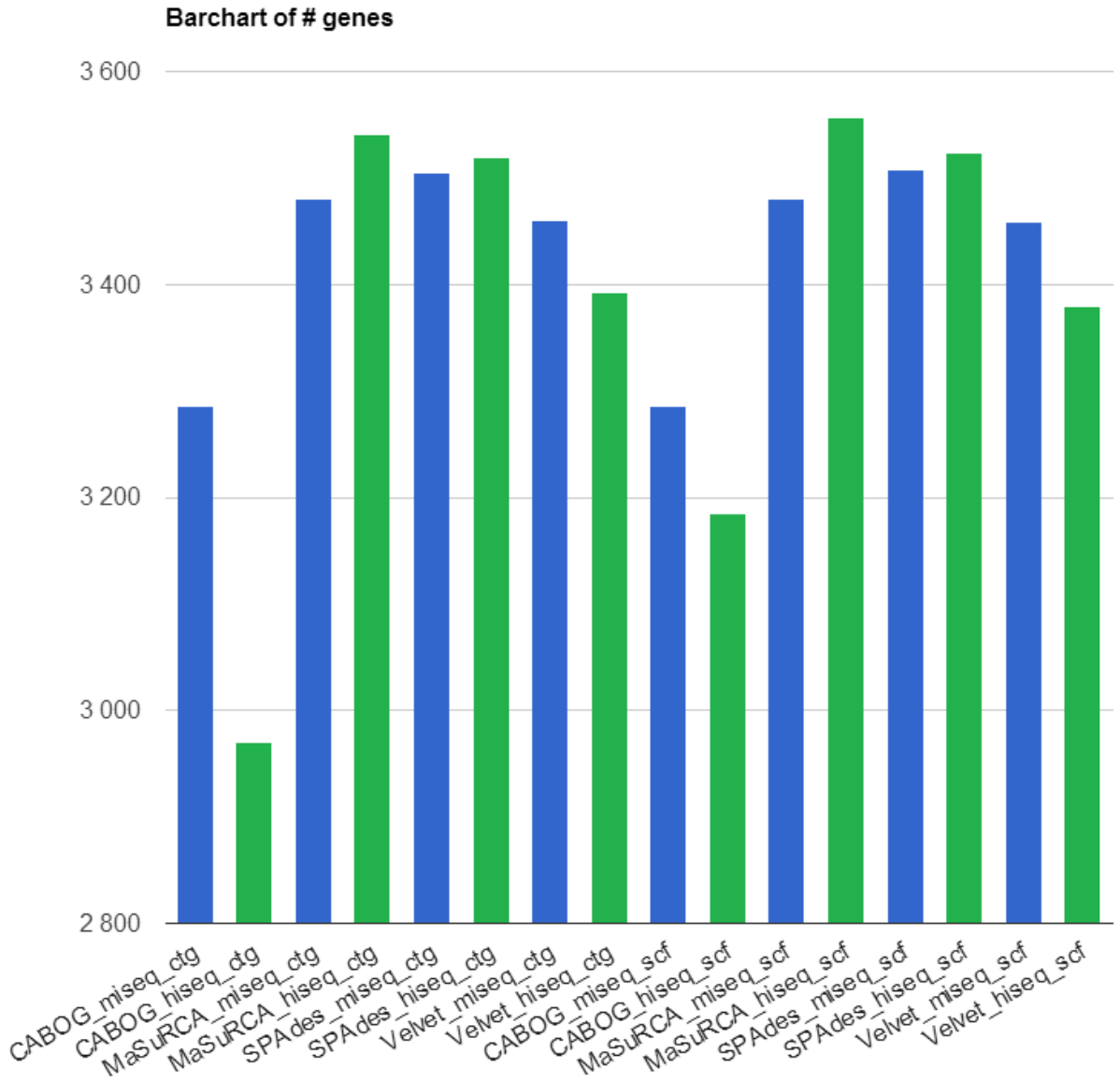


FIGURE 6-9 THE NUMBER OF GENES ON REPRODUCED RESULTS WITH NEW ASSEMBLER VERSIONS

Even though the number of genes (Figure 6-9) was reduced on assemblies performed with newer versions of SPAdes, MaSuRCA and CABOG (HiSeq data), SPAdes had the highest number of genes on assemblies with MiSeq data (Figure 6-9 blue bars), while MaSuRCA had the highest

number of genes on assemblies with HiSeq data (Figure 6-9 green bars). Somehow the new versions of MaSuRCA managed to increase the gap between their assemblies compared to SPAdes assemblies making SPAdes rank even higher as the best assembler for our bacterial genome.

6.3 ANALYSIS OF THE NEW GALAXY TOOLS FOR ASSEMBLY EVALUATION

6.3.1 PERFORMANCE

The Galaxy tools work for assemblies with both scaffold and contig files and can handle a large number of assemblies and reference genomes. The most computationally intensive part of the tool *Compute statistics* is the assessment of the assemblies performed by QUAST. The more assemblies selected as input the longer time the run will take as each input will be subject to all statistical analysis and plot-generations performed by QUAST. 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.

6.3.2 POTENTIAL USE

The Galaxy tools could be used as supplement to other assembly evaluation tools. They can be used to get comparable statistics in both text and visualized form. The tools 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 obtained 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.

6.3.3 STRENGTHS

One of the most significant strengths of the new Galaxy tools is that they contain excellent visualization features. They are easy to use and enable users to compute heavy analysis on a simple, yet reliable platform. The tools makes it easier to find differences and similarities between several assemblies based on colored tables and user-specified plot generation. The time spent on assessment of assemblies can be reduced by using these tools because users can reuse assemblies on multiple runs, easily share the results or combine statistical output without having to duplicate run sessions.

6.3.4 WEAKNESSES

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 intuitive so that the users know that it is possible to check and uncheck which assemblies to display in the plot. As it is now, this feature is far from intuitive and need to be explained explicitly 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 useful file (report.html). This can be viewed as 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. The tool Compare statistics will fail to merge two runs were one run is carried out with lower/upper threshold value X and the other is carried out with lower/upper threshold value Y, assuming that $X \neq Y$.

6.4 FURTHER WORK

Even though much has been accomplished, even more can be done. Some aspects around both the Galaxy tools for assembly evaluation and the reproducing of the GAGE-B results can be considered for further work as described in Section 6.4.1 and 6.4.2.

6.4.1 THE GALAXY TOOLS FOR ASSEMBLY EVALUATION

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 Section 6.3.4 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 they 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 Section 2.4.2, 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 desirable.

6.4.2 REPRODUCING THE GAGE-B RESULTS

Since only assemblies for *Vibrio cholerae* were tentatively reproduced with most of the assemblers used in the GAGE-B paper, a lot more can be done to fully reproduce the GAGE-B results. Further work can be to:

- Reproduce all results with the 8 assemblers on all species
- Evaluate the reproduced results
- Re-evaluate the GAGE-B conclusion based on the reproduced results
- Determine if and why there was any differences between the GAGE-B results and the reproduced results
- Rerun all species with the new versions of the assembler to observe any improvement

6.5 CONCLUSION

The new Galaxy tools for assembly evaluation proved to be quite helpful during the assessment of multiple assemblies from the GAGE-B paper. The tools offer improved visualization features in a user friendly environment that supports reproducibility and reusability of results/dataset, but they still have a long way to go before they can be considered "complete". In the end only the scientific communities contribution to expand the tools functionality and to report discovered bugs will determine the completeness of the tools.

The reproduction of the GAGE-B results turned out to be quite a challenge. This process emphasizes the importance of good documentation and consistency in the supplied data with the reported results in a paper. Unfortunately, in this case, not only was it infeasible to reproduce the results, but we were left with more questions than answers. There were too many gaps in the paper leaving the two main questions; what went wrong and why?

The reusability of some GAGE-B datasets with newer assembler versions were a bit easier to compute. With the correct installed assemblers and new Galaxy tools to evaluate the assemblies, it was easy to observe if there was any improvement or not. One assembly produced the exact same assembly as its previous version while other had improvement on some metrics, but performed worse on others.

REFERENCES

1. A. Gurevich, V. Saveliev, N. Vyahhi, et al., *Quast: Quality assessment tool for genome assemblies.* Bioinformatics, 2013. **29**(8): p. 1072-5.
2. T. Magoc, S. Pabinger, S. Canzar, et al., *Gage-b: An evaluation of genome assemblers for bacterial organisms.* Bioinformatics, 2013.
3. J. R. Miller, A. L. Delcher, S. Koren, et al., *Aggressive assembly of pyrosequencing reads with mates.* Bioinformatics, 2008. **24**(24): p. 2818-2824.
4. 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.
5. 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.
6. 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.
7. 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).
8. 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.
9. Genome assembly Available from: http://www.k.u-tokyo.ac.jp/pros-e/person/shinichi_morishita/shinichi_morishita.htm
10. W.-K. Sung, *Algorithms in bioinformatics: A practical introduction*. 2009: CRC Press (Taylor & Francis Group). 24-25.
11. M. L. Metzker, *Sequencing technologies - the next generation.* Nature Reviews Genetics, 2010. **11**(1): p. 31-46.
12. Z. Li, Y. Chen, D. Mu, et al., *Comparison of the two major classes of assembly algorithms: Overlap–layout–consensus and de-bruijn-graph.* Briefings in Functional Genomics, 2012. **11**(1): p. 25-37.
13. 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.
14. M. Pop, *Genome assembly reborn: Recent computational challenges.* Briefings in bioinformatics, 2009. **10**(4): p. 354-66.
15. A. M. Phillippy, M. C. Schatz, and M. Pop, *Genome assembly forensics: Finding the elusive mis-assembly.* Genome Biology, 2008. **9**(3).
16. Reproducibility, Access date: 08.04.14 Available from: http://todayinsci.com/K/Kohn_Alexander/KohnAlexander-Quotations.htm
17. R. D. Peng, F. Dominici, and S. L. Zeger, *Reproducible epidemiologic research.* American Journal of Epidemiology, 2006. **163**(9): p. 783-789.
18. G. K. Sandve, A. Nekrutenko, J. Taylor, et al., *Ten simple rules for reproducible computational research.* PLOS Computational Biology, 2013. **9**(10): p. e1003285.
19. E. S. Raymond, *The cathedral and the bazaar*. 1998: O'Reilly Media.
20. R. Prieto-Diaz, *Status report: Software reusability.* Software, IEEE, 1993. **10**(3): p. 61-66.
21. A. M. Bolger, M. Lohse, and B. Usadel, *Trimmomatic: A flexible trimmer for illumina sequence data.* Bioinformatics, 2014.
22. Abyss webpage, Access date: 12.06.14 Available from: http://www.bcgsc.ca/platform/bioinfo/software/abyss
23. E. W. Myers, G. G. Sutton, A. L. Delcher, et al., *A whole-genome assembly of drosophila.* Science, 2000. **287**(5461): p. 2196-2204.
24. 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.

25. A. V. Zimin, G. Marcais, D. Puiu, et al., *The masurca genome assembler*. Bioinformatics, 2013. **29**(21): p. 2669-77.
26. 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.
27. 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.
28. 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.
29. What is python, Access date: 28.05.2014 Available from: https://docs.python.org/2/faq/general.html#what-is-python
30. Os module, Access date: 28.05.2014 Available from: http://www.pythonforbeginners.com/os/pythons-os-module
31. Time module, Access date: 28.05.2014 Available from: https://docs.python.org/2/library/time.html
32. Zipfile module, Access date: 28.05.2014 Available from: https://docs.python.org/2/library/zipfile.html
33. Pypdf, Access date: 28.05.2014 Available from: https://pypi.python.org/pypi/pyPdf
34. Reportlab toolkit, Access date: 28.05.2014 Available from: https://pypi.python.org/pypi/reportlab/3.1.8
35. Reportlab user guide, Access date: 28.05.2014 Available from: http://www.reportlab.com/docs/reportlab-userguide.pdf
36. Collections module, Access date: 28.05.2014 Available from: https://docs.python.org/2/library/collections.html
37. Quast, Access date: 28.05.2014 Available from: http://quast.bioinf.spbau.ru/
38. Google chart, Access date: 28.05.2014 Available from: http://en.wikipedia.org/wiki/Google_Chart_API
39. Sqlite3, Access date: 28.05.2014 Available from: https://docs.python.org/2/library/sqlite3.html
40. Json in python, Access date: 28.05.2014 Available from: https://docs.python.org/2/library/json.html

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" (See *PCR*)
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
deoxyribonucleic acid (DNA)	A double-stranded, helical nucleic acid molecule capable of replicating and determining the inherited structure of a cell's proteins
dependency hell	a term used to define the problems faced by software developers, publishers and users in general, when software or a software package is dependent on other software
DNA	Abbreviation of *deoxyribonucleic acid*
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

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
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. The opposite is *single end reads*
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
repeats	multiple copies of the same DNA base sequence on a chromosome
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	Abbreviation for *Ribonucleic acid*
rRNA	Abbreviation 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	Abbreviation for *transport-RNA*
viral	A biological *virus*
virus	A submicroscopic, non-cellular 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
GAGE-B RECIPE

TABLE A 1 ASSEMBLER VERSIONS AND READ TYPE USED IN ASSEMBLIES FOR *VIBRIO CHOLERAE*

Raw reads are produced by the sequencer while clean (trimmed) reads are reads where adapter sequences are removed and Q10 quality trimming using the ea-utils package is performed. The trimming was performed by the GAGE-B authors.

1. The k-values in the GAGE-B recipe for MaSuRCA listed k=89 **and** k=99 for HiSeq data excluding the k values for MiSeq data resulting in runs with both k-values for both datatypes to determine the correct k-value.

Example of GapClose errors can be viewed in Table A 3

Assembler	Version in GAGE-B	Version in this thesis	Readtype for MiSeq	Readtype for HiSeq
ABySS	1.3.4	1.3.4	Clean - not finished	Clean - not finished
CABOG	7.0	7.0	Raw	Clean
		8.1	Raw	Clean
MIRA	3.4.0	3.4.0	Clean	Raw
MSRCA[1]	1.8.3	1.8.3	Clean – K89 – gapclose error[2]	Raw – K89
			Clean – K99 – gapclose error[2]	Raw – K99
		2.1.0	Clean – K89	Raw – K89
			Clean – K99	Raw – K99
SGA	0.9.34	0.9.34	Clean - not finished	Clean - not finished
SOAPdenovo2 + GapCloser	2.04 + 1.12	2.04 + 1.12	Raw	Clean
SPAdes	2.3.0	2.3.0	Clean	Clean
		2.5.0	Clean	Clean
Velvet	1.2.08	1.2.08	Clean	Clean
		1.2.10	Clean	Clean

TABLE A 2 RECIPE FOR REPRODUCTION OF *VIBRIO CHOLERAE* DATA

Assembler	Read	Command
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 *where config file contains* unitigger = bog
	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 *where config file contains* unitigger = bog
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
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

TABLE A 3 GAPCLOSE ERRORS ENCOUNTERED ON MASURCA ASSEMBLY WITH MISEQ DATA

Both k-values (89 and 99) resulted in the same error. Some absolute paths have been edited to ensure the anonymity of the server structure.

k value	Description
89	mkdir CA/10-gapclose /path/to/MSR-CA-1.8.3/bin/getEndSequencesOfContigs.perl /path/to/msrca_miseq_k89/CA/9-terminator 100 100 /path/to/MSR-CA-1.8.3/bin/create_end_pairs.perl /path/to/msrca_miseq_k89/CA/9-terminator 100 > contig_end_pairs.100.fa echo "cc 500 200" > meanAndStdevByPrefix.cc.txt jellyfish count -m 21 -t 24 -C -r -s 1 -o k_u_hash_localReadsFile_21_2_all_faux_reads contig_end_pairs.100.fa terminate called after throwing an instance of 'jellyfish::file_parser::FileParserError' what(): Empty input file 'contig_end_pairs.100.fa' child died with signal 6, with coredump
99	mkdir CA/10-gapclose /path/to/MSR-CA-1.8.3/bin/getEndSequencesOfContigs.perl /path/to/msrca_miseq_k99/CA/9-terminator 100 100 /path/to/MSR-CA-1.8.3/bin/create_end_pairs.perl /path/to/msrca_miseq_k99/CA/9-terminator 100 > contig_end_pairs.100.fa echo "cc 500 200" > meanAndStdevByPrefix.cc.txt jellyfish count -m 21 -t 24 -C -r -s 1 -o k_u_hash_localReadsFile_21_2_all_faux_reads contig_end_pairs.100.fa terminate called after throwing an instance of 'jellyfish::file_parser::FileParserError' what(): Empty input file 'contig_end_pairs.100.fa' child died with signal 6, with coredump

TABLE A 4 SOME PROBLEMS DISCOVERED IN THE GAGE-B RECIPE UPON USE

Type	Problem	Description
General	Copying some commands without errors are not feasible	Bad choice of file-format (PDF). Should use plain text format to avoid character error while copy-pasting text
CABOG	**1)** Contig file specifying unitiger = bog	**1)** Typo: unitiger → unitigger
MIRA	**1)** ”NA”	**1)** “Bad quotation mark” used. Copying this command gives character error
	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)** 'GLIBCXX_2.4.15 not found' during a run	**1)** Need to load gcc module for this run. It would be nice to know the dependencies prior to a run
	2) phred64 error on HiSeq data	**2)** Had to add --phred64 to the sga preprocess command
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
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)"). Best result for each metric is written in bold. The first data column (Supplementary table Sx) refers to data obtained from GAGE-Bs supplementary material apr4[7]. All statistics for HiSeq data is listed in GAGE-Bs Supplementary Table S6 while statistics for MiSeq data is listed in GAGE-Bs Supplementary Table S7.

The values for N50/NA50 are written in kb to make it easier to compare to the values from the GAGE-B Supplementary Tables S6-S7. The second column marked as "Assembly file" refers to assemblies downloaded from http://ccb.jhu.edu/gage_b/genomeAssemblies/index.html with various assemblers as described in Table B-2. There's a difference between two GAGE-B values (supplementary vs. assembly file) if the adjacent cells are colored light red. A cell is marked with the value N/A if there is no data available for that given metric. For the duplication ratio obtained from GAGE-Bs 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.

Each assembly contains two files used in the comparison, one with contigs and the other containing scaffolds. The only exception to this is assemblies computed by MIRA which only contains one file with contigs.

[7] http://bioinformatics.oxfordjournals.org/content/suppl/2013/05/10/btt273.DC1/GAGE-B_SupplementaryMaterial_Apr4.docx

TABLE B 1 COMPARISON OF CONTIGS FROM CABOG RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using CABOG 7.0/8.1.

Assembly	CABOG 7.0			CABOG 8.1
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# contigs (>= 200 bp)	**127**	**127**	144	519
N50 (kb)	57.9	**61.2**	57.1	10.4
NA50 (kb)	48.8	**57.8**	57.1	10.4
# misassemblies	33	20	10	**8**
# local misassemblies	12	11	**7**	**7**
# unaligned contigs	**0**	**0**	**0**	1
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

TABLE B 2 COMPARISON OF SCAFFOLDS FROM CABOG RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using CABOG 7.0/8.1.

Assembly	CABOG 7.0			CABOG 8.1
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# scaffold (>= 500 bp)	108	108	**94**	183
N50 (kb)	67.0	67.1	**134.1**	38.7
NA50 (kb)	53.2	63.2	**134.1**	38.2
# misassemblies	34	21	11	**10**
# local misassemblies	24	23	**13**	38

# 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

TABLE B 3 COMPARISON OF CONTIGS FROM CABOG RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using CABOG 7.0/8.1

Assembly	CABOG 7.0			CABOG 8.1
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# contigs (>= 200 bp)	241	241	**188**	286
N50 (kb)	32.8	**33.7**	1.7	25.0
NA50 (kb)	32.5	**33.7**	1.6	25.0
# misassemblies	22	17	**7**	8
# local misassemblies	7	7	**1**	6
# unaligned contigs	**1**	3	97	**1**
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

TABLE B 4 COMPARISON OF SCAFFOLDS FROM CABOG RUNS ON MISEQ

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using CABOG 7.0/8.1.

Assembly	CABOG 7.0			CABOG 8.1
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# scaffold (>= 500 bp)	241	241	**188**	285
N50 (kb)	32.8	**33.7**	1.7	25.0
NA50 (kb)	32.5	**33.7**	1.6	25.0
# misassemblies	22	17	**7**	9
# local misassemblies	7	7	**1**	6
# unaligned contigs	1	3	97	**1**
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

TABLE B 5 COMPARISON OF CONTIGS FROM MIRA RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with HiSeq data using MIRA 3.4.0.

Assembly	MIRA 3.4.0		
	GAGE-B		Reproduced
	Supplementary table S6	Assembly file	
# contigs (>= 200 bp)	728	**733**	1524
N50 (kb)	**92.0**	**92.0**	4.8
NA50 (kb)	87.1	**89.5**	4.8
# misassemblies	89	**24**	45
# local misassemblies	15	**9**	10
# unaligned contigs	10	**2**	7

Genome fraction (%)	**99.7**	97.925	94.917
Duplication ratio	1.0	1.016	**1.009**
# genes	N/A	**3 516**	2 627

TABLE B 6 COMPARISON OF CONTIGS FROM MIRA RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with MiSeq data using MIRA 3.4.0.

Assembly	MIRA 3.4.0		
	GAGE-B		Reproduced
	Supplementary table S6	Assembly file	
# contigs (>= 200 bp)	430	431	**224**
N50 (kb)	**112.9**	**112.9**	108.6
NA50 (kb)	**108.7**	**108.7**	108.6
# misassemblies	148	49	**23**
# local misassemblies	17	7	**4**
# unaligned contigs	20	15	**5**
Genome fraction (%)	**99.6**	98.311	98.078
Duplication ratio	1.0	1.016	**1.012**
# genes	N/A	**3 559**	3 534

TABLE B 7 COMPARISON OF CONTIGS FROM MASURCA RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last four column represents assemblies reproduced with HiSeq data using MaSuRCA 1.8.3/2.1.0 with k values 89 and 99.

Assembly	MaSuRCA 1.8.3				MaSuRCA 2.1.0	
	GAGE-B		Reproduced		Reproduced	
	Supplementary table S6	Assembly file	k=89	k=99	k=89	k=99
# contigs (>= 200 bp)	**105**	**105**	137	330	179	119
N50 (kb)	**241.6**	**241.6**	108.8	35.3	93.5	2.4
NA50 (kb)	**236.4**	**236.4**	90.9	35.3	81.5	2.2
# misassemblies	12	8	8	11	8	**2**
# local misassemblies	5	5	4	5	6	**0**
# unaligned contigs	**0**	2	2	2	4	**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

TABLE B 8 COMPARISON OF SCAFFOLDS FROM MASURCA RUNS ON HISEQ

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last four column represents assemblies reproduced with HiSeq data using MaSuRCA 1.8.3/2.1.0 with k values 89 and 99

Assembly	MaSuRCA 1.8.3				MaSuRCA 2.1.0	
	GAGE-B		Reproduced		Reproduced	
	Supplementary table S6	Assembly file	k=89	k=99	k=89	k=99
# scaffold (>= 500 bp)	**88**	**88**	**88**	245	118	167
N50 (kb)	246.5	246.5	**246.8**	46.6	133.2	65.8
NA50 (kb)	**236.4**	**236.4**	**236.4**	46.6	132.3	65.3
# misassemblies	11	**9**	10	12	13	11
# local misassemblies	8	7	15	31	20	**6**
# unaligned contigs	**0**	2	2	1	4	1
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

TABLE B 9 COMPARISON OF CONTIGS FROM MASURCA RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last four column represents assemblies reproduced with MiSeq data using MaSuRCA 1.8.3/2.1.0 with k values 89 and 99.

Assembly	MaSuRCA 1.8.3				MaSuRCA 2.1.0	
	GAGE-B		Reproduced		Reproduced	
	Supplementary table S7	Assembly file	k=89	k=99	k=89	k=99
# contig (>= 200 bp)	**173**	**173**	174	**173**	182	182
N50 (kb)	**76.1**	**76.1**	61.3	**76.1**	62.0	65.8
NA50 (kb)	71.6	**76.1**	60.5	**76.1**	61.3	65.3
# misassemblies	23	**19**	20	**19**	11	9
# local misassemblies	5	**3**	5	**3**	5	6
# unaligned contigs	**0**	3	5	3	3	1
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

TABLE B 10 COMPARISON OF SCAFFOLDS FROM MASURCA RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last four column represents assemblies reproduced with MiSeq data using MaSuRCA 1.8.3/2.1.0 with k values 89 and 99.

Assembly	MaSuRCA 1.8.3				MaSuRCA 2.1.0	
	GAGE-B		Reproduced		Reproduced	
	Supplementary table S7	Assembly file	k=89	k=99	k=89	k=99
# scaffold (>= 500 bp)	**163**	**163**	167	**163**	171	167
N50 (kb)	**76.1**	**76.1**	61.3	**76.1**	62.0	65.8
NA50 (kb)	71.6	**76.1**	60.5	**76.1**	61.3	65.3

# misassemblies	23	19	20	19	13	**11**
# local misassemblies	5	**3**	5	**3**	5	6
# unaligned contigs	**0**	3	5	3	3	1
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

TABLE B 11 COMPARISON OF CONTIGS FROM SOAPDENOVO RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with HiSeq data using SOAPdenovo2 with GapCloser.

Assembly	SOAPdenovo2 2.04		
	GAGE-B		Reproduced
	Supplementary table S6	Assembly file	
# contigs (>= 200 bp)	**139**	**139**	462
N50 (kb)	125.9	**135.1**	21.7
NA50 (kb)	106.5	**112.9**	21.7
# misassemblies	26	15	**2**
# local misassemblies	50	50	**0**
# unaligned contigs	5	2	**1**
Genome fraction (%)	**99.5**	97.295	96.488
Duplication ratio	1.0	1.009	**1.002**
# genes	N/A	**3 479**	3 274

TABLE B 12 COMPARISON OF SCAFFOLDS FROM SOAPDENOVO RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with HiSeq data using SOAPdenovo2 with GapCloser.

Assembly	SOAPdenovo2 2.04		
	GAGE-B		Reproduced
	Supplementary table S6	Assembly file	
# scaffold (>= 500 bp)	**75**	**75**	77
N50 (kb)	181.1	200.5	**200.8**
NA50 (kb)	168.1	181.1	**181.2**
# misassemblies	26	15	**6**
# local misassemblies	**76**	**76**	81
# unaligned contigs	**1**	2	**1**
Genome fraction (%)	**99.0**	97.305	97.517
Duplication ratio	1.0	**1.009**	1.011
# genes	N/A	3 480	**3 485**

TABLE B 13 COMPARISON OF CONTIGS FROM SOAPDENOVO RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with MiSeq data using SOAPdenovo2 with GapCloser.

Assembly	SOAPdenovo2 2.04		
	GAGE-B		Reproduced
	Supplementary table S7	Assembly file	
# contigs (>= 200 bp)	**244**	**244**	439
N50 (kb)	**71.4**	**71.4**	29.6
NA50 (kb)	65.5	**71.4**	29.6
# misassemblies	21	12	**2**
# local misassemblies	48	44	**0**

# unaligned contigs	4	5	**2**
Genome fraction (%)	**99.3**	96.940	96.230
Duplication ratio	1.0	1.003	**1.001**
# genes	N/A	**3 442**	3 336

TABLE B 14 COMPARISON OF SCAFFOLDS FROM SOAPDENOVO RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last column represents an assembly reproduced with MiSeq data using SOAPdenovo2 with GapCloser.

Assembly	SOAPdenovo2 2.04		
	GAGE-B		Reproduced
	Supplementary table S7	Assembly file	
# scaffold (>= 500 bp)	**165**	**165**	166
N50 (kb)	91.9	91.9	**92.1**
NA50 (kb)	89.9	91.9	**92.1**
# misassemblies	24	14	**6**
# local misassemblies	80	**77**	111
# unaligned contigs	**1**	4	2
Genome fraction (%)	**98.7**	97.050	97.068
Duplication ratio	1.0	**1.005**	**1.005**
# genes	N/A	**3 443**	3 432

TABLE B 15 COMPARISON OF CONTIGS FROM SPADES RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using SPAdes 2.30/2.5.0.

Assembly	SPAdes 2.3.0			SPAdes 2.5.0
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# contigs (>= 200 bp)	205	205	158	**134**
N50 (kb)	77.1	83.5	137.7	**225.9**
NA50 (kb)	77.1	83.5	137.7	**197.8**
# misassemblies	7	**4**	**4**	8
# local misassemblies	4	**2**	9	3
# unaligned contigs	8	**2**	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

TABLE B 16 COMPARISON OF SCAFFOLDS FROM SPADES RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using SPAdes 2.30/2.5.0.

Assembly	SPAdes 2.3.0			SPAdes 2.5.0
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# scaffold (>= 500 bp)	106	106	109	**93**
N50 (kb)	98.3	98.3	225.9	**344.0**
NA50 (kb)	94.8	95.9	214.8	**246.2**
# misassemblies	27	21	**7**	8
# local misassemblies	19	17	11	**6**

# unaligned contigs	**1**	2	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

TABLE B 17 COMPARISON OF CONTIGS FROM SPADES RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using SPAdes 2.30/2.5.0.

Assembly	SPAdes 2.3.0			SPAdes 2.5.0
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# contigs (>= 200 bp)	1475	1475	1480	**786**
N50 (kb)	**262.2**	**262.2**	**262.2**	246.5
NA50	246.6	**262.2**	**262.2**	198.5
# misassemblies	7	5	5	**4**
# local misassemblies	6	4	4	**1**
# unaligned contigs	1336	**60**	**60**	84
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

TABLE B 18 COMPARISON OF SCAFFOLDS FROM RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using SPAdes 2.30/2.5.0.

Assembly	SPAdes 2.3.0			SPAdes 2.5.0
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# scaffold (>= 500 bp)	**145**	**145**	147	184
N50 (kb)	**262.2**	**262.2**	**262.2**	258.7
NA50 (kb)	246.6	**262.2**	**262.2**	215.2
# misassemblies	7	5	5	**4**
# local misassemblies	6	4	4	**3**
# unaligned contigs	**57**	60	60	84
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

TABLE B 19 COMPARISON OF CONTIGS FROM VELVET RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using Velvet 1.2.08/1.2.10.

Assembly	Velvet 1.2.08			Velvet 1.2.10
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# contigs (>= 200 bp)	261	261	**246**	**246**
N50 (kb)	40.1	40.9	**46.3**	**46.3**
NA50 (kb)	39.5	40.9	**42.8**	**42.8**
# misassemblies	9	**4**	9	9
# local misassemblies	9	**8**	9	9
# unaligned contigs	**1**	2	**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**

TABLE B 20 COMPARISON OF SCAFFOLDS FROM VELVET RUNS ON HISEQ DATA

The values for the first column (GAGE-B supplementary table S6) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with HiSeq data using Velvet 1.2.08/1.2.10.

Assembly	Velvet 1.2.08			Velvet 1.2.10
	GAGE-B		Reproduced	Reproduced
	Supplementary table S6	Assembly file		
# scaffold (>= 500 bp)	**85**	**85**	104	104
N50 (kb)	**172.5**	**172.5**	163.4	163.4
NA50 (kb)	**171.5**	**171.5**	163.1	163.1
# misassemblies	13	**10**	**10**	**10**
# local misassemblies	132	129	**123**	**123**
# 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

TABLE B 21 COMPARISON OF CONTIGS FROM VELVET RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using Velvet 1.2.08/1.2.10.

Assembly	Velvet 1.2.08			Velvet 1.2.10
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# contigs (>= 200 bp)	201	201	**179**	**179**
N50 (kb)	92.0	92.0	**105.2**	**105.2**
NA50 (kb)	67.1	67.1	**105.2**	**105.2**
# misassemblies	12	14	**5**	**5**
# local misassemblies	7	**2**	3	3
# unaligned contigs	**1**	6	3	3
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

TABLE B 22 COMPARISON OF SCAFFOLDS FROM VELVET RUNS ON MISEQ DATA

The values for the first column (GAGE-B supplementary table S7) are obtained from GAGE-Bs supplementary material, while the next column (GAGE-B assembly file) represents data computed by running assemblies on QUAST 2.2. The last two columns represents assemblies reproduced with MiSeq data using Velvet 1.2.08/1.2.10.

Assembly	Velvet 1.2.08			Velvet 1.2.10
	GAGE-B		Reproduced	Reproduced
	Supplementary table S7	Assembly file		
# scaffold (>= 500 bp)	138	138	**133**	**133**
N50 (kb)	**110.0**	**110.0**	105.2	105.2
NA50 (kb)	92.0	75.9	**105.2**	**105.2**
# misassemblies	17	22	**6**	**6**
# local misassemblies	23	13	**6**	**6**

# unaligned contigs	**1**	4	3	3
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

TABLE B 23 INCONSISTENT GAGE-B RESULTS FOR CONTIGS

Contig results acquired from GAGE-B supplementary material (S) and by running GAGE-B assemblies on QUAST (A) for all assemblers used by GAGE-B authors on both HiSeq (H) and MiSeq (M) data. Differences are highlighted with red.

Genome fraction % for GAGE-B assemblies (A) is rounded up to 1 decimal.

Duplication ratio for GAGE-B assemblies (A) is rounded up to 1 decimal.

Assembler	# contigs	N50 (kb)	NA50 (kb)	# mis-assemblies	# local misassemblies	# unaligned contigs	Genome fraction %	Duplication ratio
ABySS S-H	206	94.5	93.0	7	18	0	99.6	1.0
ABySS A-H	206	94.5	93.0	6	17	1	97.7	1.1
ABySS S-M	267	60.5	60.3	3	0	0	99.3	1.0
ABySS A-M	267	61.0	60.5	2	0	1	96.7	1.0
CABOG S-H	127	57.9	48.8	33	12	0	96.6	1.0
CABOG A-H	127	61.2	57.8	12	11	0	95.6	1.0
CABOG S-M	241	32.8	32.5	22	7	1	97.8	1.0
CABOG A-M	241	33.7	33.7	17	7	3	97.0	1.0
MIRA S-H	728	92.0	87.1	89	15	10	99.7	1.0
MIRA A-H	733	92.0	89.5	24	9	2	97.9	1.0
MIRA S-M	430	112.9	108.7	148	17	20	99.6	1.0
MIRA A-M	431	112.9	108.7	49	7	15	98.3	1.0
MaSuRCA S-H	105	241.6	236.4	12	5	0	99.4	1.0
MaSuRCA A-H	105	241.6	236.4	8	5	2	98.1	1.0
MaSuRCA S-M	173	76.1	71.6	23	5	0	98.3	1.0
MaSuRCA A-M	173	76.1	76.1	19	3	3	97.7	1.0
SGA S-H	484	23.4	23.4	5	0	1	99.3	1.0
SGA A-H	485	23.8	23.8	3	0	1	96.3	1.0
SGA S-M	1721	27.3	27.3	109	5	6	99.6	1.2
SGA A-M	1726	27.6	27.6	2	0	3	96.6	1.0
SOAPdenovo	139	125.9	106.5	26	50	5	99.5	1.0

S-H								
SOAPdenovo A-H	139	135.1	112.9	15	50	2	97.3	1.0
SOAPdenovo S-M	244	71.4	64.5	21	48	4	99.3	1.0
SOAPdenovo A-M	244	71.4	71.4	12	44	5	96.9	1.0
SPAdes S-H	205	77.1	77.1	7	4	8	99.6	1.0
SPAdes A-H	205	83.5	83.5	4	2	2	97.4	1.0
SPAdes S-M	1475	262.2	246.6	7	6	1336	99.6	1.0
SPAdes A-M	1475	262.2	262.2	5	4	60	98.6	1.0
Velvet S-H	261	40.1	39.5	9	9	1	99.4	1.0
Velvet A-H	261	40.9	40.9	4	8	2	97.0	1.0
Velvet S-M	201	92.0	67.1	12	7	1	99.5	1.0
Velvet A-M	201	92.0	67.1	14	2	6	97.6	1.0

TABLE B 24 INCONSISTENT GAGE-B RESULTS FOR SCAFFOLDS

Scaffold results acquired from GAGE-B supplementary material (S) and by running GAGE-B assemblies on QUAST (A) for all assemblers used by GAGE-B authors on both HiSeq (H) and MiSeq (M) data. Differences are highlighted with red.

Genome fraction % for GAGE-B assemblies (A) is rounded up to 1 decimal.

Duplication ratio for GAGE-B assemblies (A) is rounded up to 1 decimal.

Assembler	# scaffold	N50 (kb)	NA50 (kb)	# mis-assemblies	# local misassemblies	# unaligned contigs	Genome fraction %	Duplication ratio
ABySS S-H	102	217.6	157.1	18	70	0	99.5	1.0
ABySS A-H	102	217.6	156.8	16	68	1	98.0	1.1
ABySS S-M	196	60.5	60.3	3	0	0	98.5	1.0
ABySS A-M	196	61.0	60.5	2	0	1	96.7	1.0
CABOG S-H	108	67.0	53.2	34	24	0	96.6	1.0
CABOG A-H	108	67.0	63.0	21	23	0	95.6	1.0
CABOG S-M	241	32.8	32.5	22	7	1	98.5	1.0
CABOG A-M	241	33.7	33.7	17	7	3	97.0	1.0
MaSuRCA S-H	88	246.5	236.4	11	8	0	99.3	1.0
MaSuRCA A-H	88	246.5	236.4	9	7	2	98.1	1.0
MaSuRCA S-M	163	76.1	71.6	23	5	0	98.3	1.0
MaSuRCA A-M	163	76.1	76.1	19	3	3	97.7	1.0
SGA S-H	331	23.4	23.4	5	0	1	96.1	1.0
SGA A-H	331	24.2	24.2	3	0	1	95.6	1.0
SGA S-M	309	27.3	27.3	4	1	0	96.4	1.0
SGA A-M	309	27.9	27.9	2	1	1	95.7	1.0
SOAPdenovo S-H	75	181.1	168.1	26	76	1	99.0	1.0
SOAPdenovo A-H	75	200.5	181.1	15	76	2	97.3	1.0
SOAPdenovo S-M	165	91.9	89.8	24	80	1	98.7	1.0
SOAPdenovo A-M	165	91.9	91.9	14	77	4	97.1	1.0
SPAdes S-H	106	98.3	94.8	27	19	1	99.6	1.0
SPAdes A-H	106	98.3	95.9	21	17	2	98.2	1.0
SPAdes S-M	145	262.2	246.6	7	6	57	99.6	1.0

SPAdes A-M	145	262.2	262.2	5	4	60	98.6	1.0
Velvet S-H	85	172.5	171.5	13	132	1	98.9	1.0
Velvet A-H	85	172.5	171.5	10	129	1	97.1	1.0
Velvet S-M	138	110.0	92.0	17	23	1	99.2	1.0
Velvet A-M	138	110.0	75.9	22	13	4	97.6	1.0

TABLE B 25 COMPARISON OF CONTIGS REPRODUCED WITH THE ASSEMBLER VERSIONS USED IN GAGE-B PAPER

Assemblies reproduced with ABySS and SGA is excluded from the comparison based on the unsuccessfull runs as described earlier in the thesis.

Read type	Assembler	#contigs ≥200bp	N50 (kb)	NA50 (kb)	#mis-assemblies	#local misassemblies	# unaligned contigs	Genome fraction	Duplication ratio	#genes
MiSeq	CABOG	188	1.7	1.6	7	1	97	7.639	1.011	123
	MIRA	224	108.7	108.7	23	4	5	98.078	1.012	3 534
	MaSuRCA	**173**	76.1	76.1	19	3	3	97.670	1.023	3 534
	SOAPdenovo	439	29.6	29.6	**2**	**0**	**2**	96.230	**1.001**	3 336
	SPAdes	1 480	**262.2**	**262.2**	5	4	60	**98.752**	1.004	**3 597**
	Velvet	179	105.2	105.2	5	3	3	96.234	1.007	3 460
HiSeq	CABOG	144	57.1	57.2	10	7	**0**	95.361	1.006	3 346
	MIRA	1 524	4.8	4.8	45	10	7	94.917	1.009	2.627
	MaSuRCA	**137**	108.8	90.9	8	4	2	98.121	1.008	3.552
	SOAPdenovo	462	21.7	21.7	**2**	**0**	1	96.488	**1.002**	3 274
	SPAdes	158	**137.7**	**137.7**	4	9	3	**98.611**	1.007	**3 571**
	Velvet	246	46.3	42.8	9	9	1	96.340	1.012	3 392

TABLE B 26 COMPARISON OF SCAFFOLDS REPRODUCED WITH THE ASSEMBLER VERSIONS USED IN GAGE-B PAPER

Assemblies reproduced with ABySS and SGA is excluded from the comparison based on the unsuccessfull runs as described earlier in the thesis.

Read type	Assembler	#scaffolds ≥500bp	N50 (kb)	NA50 (kb)	#mis-assemblies	#local misassemblies	# unaligned contigs	Genome fraction	Duplication ratio	#genes
MiSeq	CABOG	188	1.7	1.6	7	**1**	97	7.639	1.011	123
	MaSuRCA	163	76.1	76.1	19	3	3	97.670	1.023	3 534
	SOAPdenovo	166	92.1	92.1	6	111	**2**	97.068	1.005	3 432
	SPAdes	147	**262.2**	**262.2**	**5**	4	60	**98.752**	**1.004**	**3 598**
	Velvet	**133**	105.2	105.2	6	6	3	96.242	1.007	3 459
HiSeq	CABOG	94	134.1	134.1	11	13	**0**	95.321	1.023	3 383
	MaSuRCA	88	**246.8**	**236.4**	10	15	2	98.152	1.013	3 568
	SOAPdenovo	**77**	200.8	181.2	**6**	81	1	97.517	**1.011**	3 485
	SPAdes	109	225.9	214.8	7	**11**	3	**98.753**	1.062	**3 586**
	Velvet	104	163.4	163.1	10	123	1	96.251	1.016	3 379

TABLE B 27 COMPARISON OF CONTIGS REPRODUCED WITH THE NEW ASSEMBLER VERSIONS

Assemblies reproduced with ABySS and SGA is excluded from the comparison based on the unsuccessfull runs as described earlier in the thesis. Other than that, reproduced results with all new assemblers (exept MIRA) when possible are provided below.

Assembly	MiSeq				HiSeq			
	CABOG 8.1	MaSuRCA 2.1.0	SPAdes 2.5.0	Velvet 1.2.10	CABOG 8.1	MaSuRCA 2.1.0	SPAdes 2.5.0	Velvet 1.2.10
# contigs (>= 200 bp)	286	182	786	**179**	519	179	**134**	246
N50 (kb)	25.0	65.8	**246.5**	105.2	10.4	93.5	**225.9**	46.3
NA50 (kb)	25.0	65.2	**198.5**	105.2	10.4	81.5	**197.8**	42.8
# misassemblies	8	9	**4**	5	**8**	**8**	**8**	9
# local misassemblies	6	6	**1**	3	7	6	3	9
# unaligned contigs	**1**	**1**	84	3	**1**	4	3	**1**
Genome fraction (%)	93.765	96.349	**97.350**	96.234	91.790	**98.498**	97.468	96.340
Duplication ratio	1.009	1.027	**1.001**	1.007	**1.002**	1.009	1.006	1.012
# genes	3 286	3 480	**3 505**	3 460	2 970	**3 541**	3 519	3 392

TABLE B 28 COMPARISON OF SCAFFOLDS REPRODUCED WITH THE NEW ASSEMBLER VERSIONS

Assemblies reproduced with ABySS and SGA is excluded from the comparison based on the unsuccessfull runs as described earlier in the thesis. Other than that, reproduced results with all new assemblers (exept MIRA) when possible are provided below.

Assembly	MiSeq				HiSeq			
	CABOG 8.1	MaSuRCA 2.1.0	SPAdes 2.5.0	Velvet 1.2.10	CABOG 8.1	MaSuRCA 2.1.0	SPAdes 2.5.0	Velvet 1.2.10
# scaffold (>= 500 bp)	285	167	184	**133**	183	118	**93**	104
N50 (kb)	25.0	65.8	**258.7**	105.2	38.7	133.2	**344.0**	163.4
NA50 (kb)	25.0	65.3	**215.2**	105.2	38.2	132.3	**246.2**	163.1
# misassemblies	9	11	**4**	6	10	13	**8**	10
# local misassemblies	6	6	**3**	6	38	20	**6**	123
# unaligned contigs	**1**	**1**	84	3	**0**	4	3	1
Genome fraction (%)	93.765	96.353	**97.356**	96.242	92.029	**98.498**	97.478	96.251
Duplication ratio	1.009	1.027	**1.001**	1.007	1.044	**1.010**	1.011	1.016
# genes	3 286	3 480	**3 508**	3 459	3 185	**3 557**	3 524	3 379