Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Borba Neumann

A Framework Approach for Quality Feature Analysis of Genome Assemblies

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Sérgio Lifschitz

Rio de Janeiro
June 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Borba Neumann

A Framework Approach for Quality Feature Analysis of Genome Assemblies

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Sérgio Lifschitz
Advisor
Departamento de Informática – PUC-Rio

Profª. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Prof. Antonio Basilio de Miranda
Fundação Oswaldo Cruz – Fiocruz

Prof. Ruy Luiz Milidiu
Departamento de Informática – PUC-Rio

Rio de Janeiro, June 24th, 2019

Guilherme Borba Neumann

He is bacharel in Biological Sciences by PUC-Rio and presentes a great interest in the field of Bioinformatics and Genomics, with experience in Genome Assembly and DNA Sequencing. He also has technical training in Telecommunications by CEFET / RJ.

Bibliographic Data

Neumann, Guilherme Borba

A Framework Approach for Quality Feature Analysis of Genome Assemblies / Guilherme Borba Neumann; advisor: Prof. Sérgio Lifschitz. – Rio de janeiro: PUC-Rio , Departamento de Informática, 2019.

v., 79 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Montagem de Genomas;. 3. Análise de Features;. 4. Métricas de Avaliação;. 5. Genômica;. 6. Análise de Qualidade;.
I. Lifschitz, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Firstly, I am really thankful for all who have motivated me to start a Master in Informatics. It was such a great experience, and now everything begins to become clearer in my career goals.

Many thanks to BioBD students, and master student colleagues, in listening to me, criticizing, and sharing hard and good times.

Personal and Professional thanks to my advisor, Sergio Lifschitz, the first one to believe me, and to convince me, that a good biologist, may be a great bioinformatician. Thank you for helping me going beyond.

I acknowledge the big support I received from Fernanda Baiao and Marcos Catanho, and from the jury, Simone Junqueira, Ruy Milidiu, and Antonio Basilio. Your attention and interest were crucial to me.

Finally, special thanks to close people. First Ema and Livia, by often good lab discussions. Friends and Family, that always gave me emotional support to keep all up. And to my dear husband, who helped emotionally, and actively in parallel work, giving me more time to study.

I am thankful for receiving this opportunity from God, to study for long 6 years at PUC, and now, having the grace to become a Master of Science and going abroad as a scientist.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001.

Abstract

Neumann, Guilherme Borba; Lifschitz, Sérgio (Advisor). **A Framework Approach for Quality Feature Analysis of Genome Assemblies**. Rio de Janeiro, 2019. 79p. MSc Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The Genome Assembly research area has quickly evolved, adapting to new sequencing technologies and modern computational environments. There exist many assembler software that consider multiple approaches. However, at the end of the process, one can always question the quality of assemblies. When an assembly is accomplished, some quality features may be generated, in order to qualify it. Nonetheless, the features do not directly tell one about assembly quality, but only bring to the biologists quantitative assembly descriptions. We propose a Domain Framework for the feature analysis process post-genome Assembly. Our goal is to enable data interpretation and assembly quality evaluation. The Genome Assembly Analysis Framework (GAAF) was designed to work with distinct species, assemblers and features. In order to validate our proposal, we have run a few practical experiments with GAAF, which make us understand the way it can be used, instantiated and extended.

Keywords

Genome Assembly; Feature Analysis; Evaluation Metrics; Genomics; Quality Analysis;

Resumo

Neumann, Guilherme Borba; Lifschitz, Sérgio. **Uma abordagem de Framework para Análise de Medidas de Qualidade da Montagem de Genomas** . Rio de Janeiro, 2019. 79p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A área de pesquisa em Montagem de Genomas tem evoluído rapidamente, adaptando-se às novas tecnologias de sequenciamento e modernos ambientes computacionais. Existem diversos softwares montadores que usam múltiplas abordagens, porém persiste o questionamento sobre a qualidade da montagem ao final do processo. Assim que uma montagem é finalizada, muitas medidas de qualidade podem ser geradas, a fim de que a montagem seja qualificada. Todavia, essas medidas apenas fornecem aos biólogos valores quantitativos acerca da montagem. Nós propomos nesta pesquisa um framework de domínio para o processo de análise de medidas pós montagem de genomas. Nosso objetivo é de prover a interpretação dos dados e avaliação da qualidade das montagens a partir do Framework. O Genome Assembly Analysis Framework (GAAF) foi projetado para trabalhar com espécies, montadores e medidas distintas. Para validar nossa proposta, foram realizados testes com o GAAF que permitem entender como o mesmo pode ser utilizado e de que maneira ele pode ser instanciado e/ou estendido.

Palavras-chave

Montagem de Genomas; Análise de Features; Métricas de Avaliação; Genômica; Análise de Qualidade;

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BUSCO – Benchmarking Universal Single Copy Orthologs

DBG – de Bruijn Graph

FRCurve – Feature-Response Curve

GAAF – Genome Assembly Analysis Framework

GAGE – Genome Assembly Gold Standard Evaluations

ICA – Independent Component Analysis

NCBI – National Center for Biotechnology Information

NGS – Next-Generation Sequencing

OLC – Overlap Layout Consensus

PCA - Principal Component Analysis

QUAST – QUality ASsessment Tool

REAPR – Recognition of Errors in Assemblies using Paired Reads

WGS – Whole Genome Sequencing

"[...] knowledge of sequences could contribute much to our understanding of living matter."

Frederick Sanger, *1980.*

1
Introduction: The Problem of Quality

We have been living a considerable revolution in Life Sciences since the advent of Information Technology – the Era of data, technology, and computation. In Genomics, it is not different. As technology evolves, data structure changes and computation needs to respond to this evolution.

In the field of Genome Sequencing and Genome Assembly, we currently face the problem of quality. The sequencing technology available nowadays generates a significant number of pieces of DNA that are submitted to assembly algorithms, in order to create an assembly as close as possible to the original molecule.

However, we claim that the distance between real genomes and assemblies is still an open issue in the area of Genome Assembly. Since high-quality assemblies are expensive and hard to achieve, when applications do not strictly require that, scientists leave assemblies at a level where they are called drafts – they are not completely finished, in comparison to reference genomes.

In GenBank[1], more than 80% of all bacterium genomes are considered drafts (Land et al. 2014). Analyzing the Genomes OnLine Database (GOLD)[2], we found 121,000 permanent drafts, in contrast to only 4,000 complete assemblies (on 14th June 2018).

But to say whether it is a problem, we need to know the quality of the drafts. On the one hand, Denton et al. have found extensive errors in the number of genes inferred from draft genome assemblies, mainly due to fragmentation of genes (Denton et al. 2014). On the other hand, Land et al. reported that 88% of all GenBank bacterium draft assemblies are good enough according to some thresholds in contiguity and base analysis (Land et al. 2014). For example, genome annotation, during Synteny Analysis (to an error rate below 5%), requires a minimum N50 of 200kb and 1Mb when gene density is 290 and 200 genes per Mb, respectively (Liu et al. 2018).

[1] GenBank is the NIH (National Institutes of Health) genetic sequence database
[2] https://gold.jgi.doe.gov/

1.1 First Sequencing Generation

In only two decades, the modern biology has reformed whole science, after human genome project conclusion in 2001 (Consortium 2004) — the project spent more than ten years to find out what today is done in some weeks or even days. Today, almost all research areas are influenced by Genetics and Genomics, such as Energy, Agroindustry, Medicine and Engineering.

The first complete sequenced genome was from Bacteriophage MS2, done in 1976 (Sanger et al. 1977a). The technology available that moment was based on "plus and minus" method (Sanger and Coulson 1975), a variant of the Sanger methodology to sequence DNA, in which deoxyribonucleotides (dNTPs) are used in different reactions to generate assorted length sequences, fractioned later by gel electrophoresis. That chemical sequencer was able to generate a whole DNA fragment and was responsible for the beginning of the Bioinformatics Era.

Frederick Sanger kept improving his technology, creating in 1977 the famous chain-termination or dideoxy technique (Sanger et al. 1977b), which is still used in many places until current days. It uses dideoxynucleotides (ddNTPs), dNTPs analogs that lack 3'hydroxyl group — required for DNA extension during its synthesis. Radiolabelled ddNTPs are mixed in four parallel synthesis reactions to generate the original sequences through an autoradiography.

Several other changings had been made to improve this technique, using phospho- or tritrium- radiolabeling with fluorometric based detection or detection through capillary based electrophoresis (Heather and Chain 2016). However, the machines could not produce reads more than one kilobase in length, the reason why shotgun sequencing technique was developed later, in order to assemble those reads into long contiguous sequences (contigs), by a number of cloned and separately sequenced overlapping DNA fragments.

In addition, the creation of technologies such as polymerase chain reaction (PCR) (Saiki et al. 1988) and recombinant DNA (Jackson et al. 1972) made possible a much more quantity of pure DNA to sequence.

1.2 Second Sequencing Generation

New technologies (Shendure and Ji 2008) appeared using the luminescent method (Nyrén and Lundin 1985) for measuring pyrophosphate synthesis (Ronaghi et al. 1998): pyrosequencing was licensed to 454 Life Sciences, the first big company to launch next-generation sequencing (NGS) technology. A

great transition was made, when they started mass parallelization of sequencing reactions, increasing the amount of DNA – producing millions of 400-500 base pairs (bp) long reads (Heather and Chain 2016).

Techniques comparable to 454 emerged in the following years, among them, Solexa, later acquired by Illumina, using 'bridge amplification' method (Fedurco et al. 2006). Although, the first Genome Analyzer (GA) machine (by Solexa) was capable of generating very short reads – about 35bp long –, it could produce paired-end (PE) data (forward and reverse DNA information), improving the accuracy at mapping reads to a reference genome (Heather and Chain 2016). The second GA version was later replaced by HiSeq/MiSeq, with longer read lengths – 150bp long (Quail et al. 2012).

Another important company was Applied Biosystems (Life Technologies merged with Invitrogen, currently Thermo Fisher Scientific), owner of SOLiD (McKernan et al. 2009), a ligation and detection sequencing system. SOLiD was followed by Ion Torrent, a platform in which nucleotide incorporation is detected by the difference in pH, caused by the release of protons during DNA synthesis (Rothberg et al. 2011) — it can generate 200bp long reads (Quail et al. 2012). However, interpreting homopolymer sequences is not an easy task in Ion Torrent, due to the loss of signal of many simultaneous dNTP incorporation (Loman et al. 2012).

The sequencing cost has been dramatically altered by these companies, revolutionizing the complexity of microchips and increasing the number of chemical methods to sequence (Heather and Chain 2016). Illumina, though, has been considered the most successful sequencing platform, making this company a near monopoly (Greenleaf and Sidow 2014; Heather and Chain 2016).

1.3 Next Generation Sequencing

Currently, we are living Third-generation DNA sequencing (Schadt et al. 2010; Heather and Chain 2016), a step into longer reads, real-time sequencing, and new technologies. These technologies can sequence single molecules lacking DNA amplification, needed in all previous sequencers (Heather and Chain 2016).

A first Single Molecule Sequencing (SMS) machinery was commercialized by Helicos BioSciences (Harris et al. 2008), working with the same methodology Illumina is used to do, but with no bridge amplification — it avoids biases and errors associated to amplified DNA.

But now, one of the most famous third-generation sequencings is the Single Molecule Real-Time (SMRT) technology from Pacific Biosciences (PacBio).

Despite the cost, PacBio has been used to generate much longer reads, up to 10kbase (van Dijk et al. 2014), necessary to assemble big genomes – as the 32-gigabase-pair axolotl genome, the biggest genome ever assembled at the time of writing (Nowoshilow et al. 2018) —, although, high base detection error is still an issue to settle.

Nanopore technologies have also appeared as a promise to the future of sequencing (Haque et al. 2013). The firsts nanopore sequencers were developed by Oxford Nanopore Technologies (ONT) – GridION and MinION (Clarke et al. 2009; Eisenstein 2012) -–, and the latter were innovated by size, similar to a USB drive (Loman and Quinlan 2014). Nanopore sequencers are hoped to be a future solution to fast, low-cost and compact machines with long and accurate reads (Heather and Chain 2016). For the moment, they can be used in association with current accurate technologies due to their long reads (Karlsson et al. 2015; Madoui et al. 2015).

1.4 Genome Assembly Strategies

After sequencing, genome assembly is needed, even when using Sanger platforms. However, when one hears about the Genome Assembly, it may be related to two different concepts: *de novo* and mapping. *De novo* genome assembly intends to reconstruct DNA molecules in the absence of a reference genome (an NP-hard problem). The Mapping or Alignment approach uses a sequenced genome from the same or related species as a guide during assembly – making it much easier when comparing to *de novo* genome assembly (Pop 2009; Miller et al. 2010).

Many strategies to *de novo* assembly have already been proposed, and are basically divided into three groups: Greedy, Overlap-Layout-Consensus (OLC) and De Bruijn Graph (DBG); which are summarized as follows. String Graph Algorithm (SGA) (Myers 2005) is not as usual as the other strategies, and will not be discussed here. Let us define first, read, contig and scaffold.

Read is one sequence read/generated by the sequencing machine.

Contig derives from the word contiguous, and means a set of overlapping DNA fragments (reads) that together produce a consensus region of DNA (Staden 1980).

Scaffold is a series of contigs separated by gaps of known length.

1.4.1 Greedy Assembly Algorithms

Similarly to any other greedy algorithm, greedy assembly algorithms select each operation the highest-scoring read/contig overlap, according to some order; depending on the software (Miller et al. 2010). Therefore, contigs and later on, scaffolds, are assembled as large as possible. The Greedy Approach was widely used for assembling Sanger data, but also for NGS in assemblers such as SSAKE (Warren et al. 2007), SHARCGS (Dohm et al. 2007) and VCAKE (Jeck et al. 2007).

1.4.2 Overlap-Layout-Consensus (OLC)

Alike greedy algorithm, a list of highest-scoring overlap to each read is given in OLC (Staden 1980). The list is used for creating an overlap graph, in which each read corresponds to a node, connected by edges that represent an overlap between the corresponding nodes (Figure 1.1).

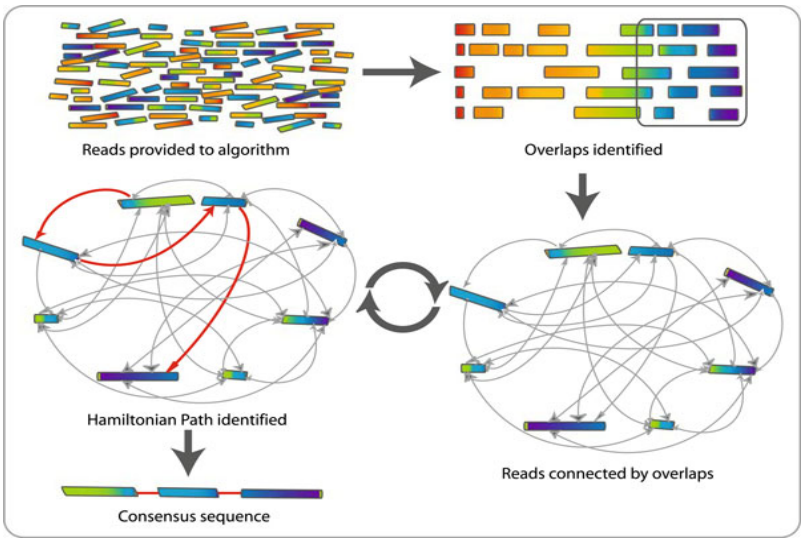

Figure 1.1: The identified overlaps list is generated from input reads. The overlaps are used as edges connecting the reads, as nodes. So, the graph created is used to find the best path which reconstructs the whole genome through Hamiltonian path algorithm. Figure from (Commins et al. 2009)

A layout step is responsible for identifying paths throughout the graph in order to generate genome fragments, or contigs. The ideal path would traverse each node in the graph only once, reconstructing the whole genome (Pop 2009). Finding this path is computationally difficult, an NP-hard problem, known as the Hamiltonian Path. The overlap strategy has time complexity $O(n^2)$ (Chen et al. 2017).

1.4.3
de Bruijn Graph Algorithm (DBG)

De Bruijn is another graph approach, widely used for short reads, that implements K-mer[3] strategy (Idury and Waterman 1995). In de Bruijn graph, k-mers are nodes, and exactly overlapping of length k - 1 between two adjacent nodes are edges; each repeat is presented at once in the graph, with links to different start and end positions (Zerbino and Birney 2008).

A path in the graph is found using a Eulerian path algorithm (O(n)). There are several efficient algorithms for finding the Eulerian path, perhaps it can generate an exponential number of Eulerian paths (Pop 2009). In addition, finding a Hamiltonian path may be reduced to finding a Eulerian path in a (k-1)-mer DBG (Chen et al. 2017).

However, a problem from k-mer approach leads to a loss of information — "long-range connectivity information implied by each read" (Pop 2009). To incorporate read information, Pevzner and colleagues (2001) created a Eulerian path variation, called Eulerian Superpath Problem (Pevzner et al. 2001). This superpath is produced from sub-paths corresponding to given reads.

The definition of K is an important decision when using DBGs. Compared to read length, a smaller k-mer can improve the overlap between strings and decrease the number of edges in the graph for storage. But also, it can generate too many nodes, unfailing genome assembly. K-mers shorter than some repeat regions (we discuss repeats in the next section) can cause graph disorder and break up the contigs (Chen et al. 2017). Thereby, a longer k might be desired but limited by read length – "if the overlapping region of two reads is less than k characters, they do not have any common vertex in the graph" (Chen et al. 2017).

All these concepts present in the assembly strategies will affect the final quality. For that reason, we see many of them as descriptors or features of assembly quality. We discuss that in Chapter 2.

1.5
Repetitive DNA

Another point that comes to mind when discussing genome assembly, is undoubtedly repetitive DNA – very similar or identical sequences present in the same genome (Treangen and Salzberg 2012). Genomes are filled with several repeat regions. The Human genome, for example, is about 50% composed of repetitive regions (Batzer and Deininger 2002).

[3]Oligomers of a given length K generated from sequenced reads

Basically, it is difficult for an assembler to differentiate a sequencing error from a polymorphism between almost identical repeat copies, and to detect the correct positions of identical repeats (Phillippy et al. 2008).

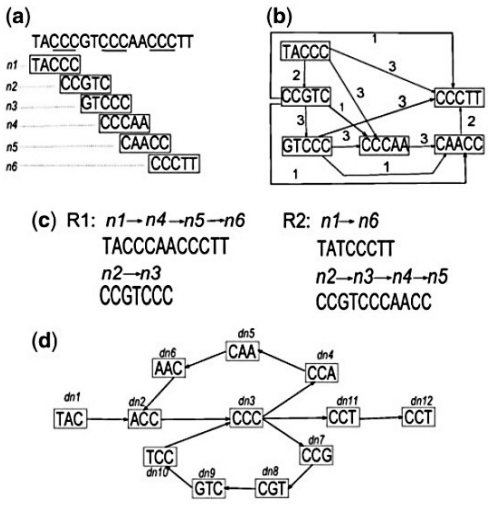

Figure 1.2: Biases from repeat regions. Figure taken from (Chen et al. 2017). (A) The sequencing machine detects six reads, namely, n1, n2, n3, n4, n5 and n6 derived from the DNA sequence 'TACCCGTCCCAACCCTT'. The DNA sequence has three repeat regions (CCC) underlined. (B) Constructing the overlapping graph with the six reads. (C) We can obtain two results (R1, R2) by using the greedy algorithm, as there are two strategies that traverse the map. From the observation, the greedy approach cannot obtain the correct sequence and fails to determine the best result. Also, these reads are assumed to come from two DNA sequences, namely, 'TACCCAACCTT' and 'CCGTCCC'. Two results (R1 and R2) are generated, so it is hard to decide the correct sequence. Therefore, the repeat regions can generate ambiguous results and lead to erroneous assembly. (D) Constructing six reads using the de-Bruijn graph. The length of the k-mer is 3, and the repeat region (CCC) is presented by the node dn3.

On the one hand, biologically, repeats represent an important feature to study, to cite some: polyploidy, satellite DNAs[4], transposons[5] (44% of the human genome), and duplicated genes. Repetitive sequences may be in numbers of 10, 1,000 or even 10^5 copies in the genome; often interleaved with single copy sequences (Snustad and Simmons 2012, p.240). In addition, high demand proteins, such as actin and myosin, or ribosomal proteins, can be

[4]Repetitive short sequences disposed in *tandem* as part of groups are named satellite DNAs. Depending on the length we can call them Microsatellites or Minisatellites. The quantity is really variable. In mammals they represent <10% of the genome and in *Drosophila virilis* around 50% (Lewin 2009).

[5] Transposable elements or transposons are mobile sequences in the genome that may change their position, generating new DNA copies.

codified by several distinct genes, genes which are moderately repetitive. The same occurs with ribosomal RNAs (Snustad and Simmons 2012, p.240).

On the other hand, computationally, repeats create a technical challenge, since they appear as ambiguities in alignment and assembly, producing biases and errors in the final sequenced genome (Treangen and Salzberg 2012). A Greedy algorithm, for example, fails to detect repeats (Figure 1.1 C), since it looks for the highest overlapping score (Chen et al. 2017). Substrings of length k, the k-mer strategy, may handle the repeat regions, but still with some errors. Figure 1.1 D shows that the de-Bruijn graph generates more than one answer with the correct contig between them.

1.6 Genome Annotation

A final aspect to consider in the context of quality analysis is an assembly post-process called Genome Annotation. There are many other applications to a Genome Assembly, so read Genome Annotation as a main example. Genome annotation is a non-standardized procedure that essentially describes genes and their associated protein or RNA products, focused on their functions (Koonin and Galperin 2003).

This process of annotation is really challenging because it is strictly related to contiguity and correctness (base-level reliability). Both in protein-coding genes and in non-coding RNAs gene finding and annotation are confronted with errors in the assembly step and to some inherent biological characteristics of species, like repeat patterns. Then, wrong gene assembly, gene fragmentation (through distinct contigs), repeats and/or duplicated genes omission, misassemblies, or genes inversion etc., may affect a correct annotation (Yandell and Ence 2012).

At this point, threshold quality metrics, such as those proposed by Liu et al. (2018), may help to mitigate the problem of published bad quality drafts and consequently incorrect annotations (Liu et al. 2018).

The next chapter will focus on listing some strategies to qualify assemblies, in the interest of creating a better understanding of drafts' quality. Chapter 3 describes our proposed solution to study quality features from genome assemblies, and Chapter 4 tests this solution in a study case. The last chapter summarizes our contributions and explains the next steps.

2
Assembly Quality: a Systematic Mapping Study

> "Too often, assembly quality is judged only by contig size, with larger contigs being preferred. However, large contigs can be the result of haphazard assembly and are not a good measure of quality. It has been difficult to gauge assembly quality by other means, because no automated validation tools exist."
>
> (Phillippy et al. 2008, p.1)

Ten years ago, Phillippy complained about the way scientists were used to qualifying assemblies. His work has motivated many others since then, creating and recreating assembly evaluation metrics. In order to go deeper into the problem of assembly quality, and to have a better description of the literature, we conducted a Systematic Mapping on the Scopus database.

2.1
Methodology

In the Scopus database, we used the following search terms:

TITLE-ABS-KEY (genome AND assembly AND ((evaluation OR assessment OR quality) AND analysis) AND (de AND novo))[1].

We selected papers through exclusion and inclusion criteria, aiming to list the approaches proposed in the last 10 years to evaluate an assembly. We also applied backward snowball sampling to include related works.

Criteria to include an article:
1. The paper presented a new genome assembly evaluation strategy;
2. The assembly is not reference-guided;

[1] Applying the query TITLE-ABS-KEY ("genome assembly" AND (validat* OR evaluat* OR assess* OR "quality analysis") AND (metric OR measure*) AND "de novo") we found only 26 papers, and using the query TITLE-ABS-KEY ("genome assembly" AND (validat* OR evaluat* OR assess* OR "quality analysis") AND "de novo") , 192 works. Both queries lacked classical papers, such as Assemblathon 1. For this reason, we chose to use a more general search.

3. The assembly evaluation method was based on a validated methodology or validated its own methodology;
4. The paper was published from 2008 to 2018.

Criteria to exclude an article:
1. The paper was not written in English;
2. The study was a book or gray literature (posters, summaries of articles, tutorials, and panels);
3. Metagenomics or transcriptome study[2].

Firstly, we employed the criteria on title, abstract and year of publishing. Secondly, the remaining articles were selected according to full-text, based on the same inclusion and exclusion criteria. We collected data such as title, authors' name, year of publication, journal's name, proposed evaluation metrics, and tools.

2.2 Results

The search returned 623 papers, which were filtered by type (resulting in 610) and by language (605). Transcriptome studies (NOT (transcriptome OR transcriptomic OR transcriptional)) were removed, and 245 articles remained. Applying the inclusion and exclusion criteria on title and abstract, we chose 25 studies. After full-text reading, we selected 12 articles through inclusion and exclusion criteria and 3 more through backward snowball sampling (QUAST, BUSCO and Genome Assembly Forensics). All papers are listed in Table 2-1. The papers are ordered by the number of citations.

On the one hand, the most cited work in this area, QUAST, was developed in 2013 to gather the most widely used metrics into a single tool (http://quast.sourceforge.net). The software presents the metrics of the Assemblathon competition (Earl et al. 2011) and GAGE (Salzberg et al. 2012), and some modified Nx[3] metrics (Gurevich et al. 2013). QUAST is currently integrated with Icarus (Mikheenko et al. 2016), a visualizer for assembly contigs. QUAST is fundamentally based on Plantagora (Barthelson et al. 2011), an out-of-date web-based plant genome assembly evaluation platform, and on GAGE. It also integrates met-

[2]Out of scope

[3] Considering a decreasing-ordered list of contigs, Nx (e.g., N50, N90) is the length of the shortest contig from the sum group of all contigs from the list necessary to get x% of total assembly length. NGx considers not the total assembly length, but the original genome length. And NAx does the same job as Nx but using an aligned contigs list; contigs containing misassemblies are broken into two new contigs (Gurevich et al. 2013).

Table 2.1: Systematic Mapping Results ordered according to the number of citations (by Google Scholar on 16th May 2018)

Author	**Title**	**Year**	**Citations**
A.Gurevich*et al.*	QUAST: quality assessment tool for genome assemblies	2013	947
F. A. Simão*et al.*	BUSCO:assessing genome assembly and annotation completeness with single-copy orthologs	2015	892
D. Earl*et al.*	Assemblathon 1: A competitive assessment of de novo short read assembly methods	2011	404
K.R. Bradnam*et al.*	Assemblathon 2: Evaluating de novo methods of genome assembly in three vertebrate species	2013	391
A. M. Phillippy, M. C. Schatz, M. Pop	Genome assembly forensics: finding the elusive missassembly	2008	240
M. Hunt*et al.*	REAPR: A universal tool for genome assembly evaluation	2013	210
G. Narzisi, B. Mishra	Comparing De Novo genome assembly: The long and short of it	2011	131
A. Rahman, L. Pachter	CGAL: Computing genome assembly likelihoods	2013	68
F. Vezzi, G. Narzisi, B. Mishra	Feature-by-feature - evaluating De Novo sequence assembly	2012	60
F. Vezzi, G. Narzisi, B. Mishra	Reevaluating Assembly Evaluations with Feature Response Curves: GAGE and Assemblathons	2012	57
A.E. Darling*et al.*	Mauve assembly metrics	2011	53
A. Mikheenko*et al.*	Icarus: Visualizer for de novo assembly evaluation	2016	4
C. Lo, S. Kim, S. Zakov, V. Bafna	Evaluating genome architecture of a complex region via generalized bipartite matching	2013	4
C.J. Castro, T.F.F. Ng	U50: A New Metric for Measuring Assembly Output Based on Non-Overlapping, Target-Specific Contigs	2017	1
J. Abante*et al.*	HiMMe: Using genetic patterns as a proxy for genome assembly reliability assessment	2017	1

rics from GeneMark.hmm (Lukashin and Borodovsky 1998) and GlimmerHMM (Majoros et al. 2004), introducing, in addition, NA50.

On the other hand, the second most cited paper, Benchmarking Universal Single Copy Orthologs (BUSCO), utilizes a distinct strategy to validate/evaluate an assembly (Simão et al. 2015). BUSCO is a software that provides quality metrics for genome assembly based on ortholog relationships. The metrics C:complete, D:duplicated, F:fragmented, M:missing and n:number of genes used are generated comparing ortholog genes identified in the assembly against genes available at their database[4].

We could not forget one of the most classic works in this field of Genome Assessment: the Assemblathon Competition. In its editions 1 and 2, the event presented a series of metrics to qualify an assembly with the intention of using assemblies' quality as an assembler validation process. Assemblathon presents more than 100 metrics, classified in groups, such as coverage, contiguity, structure, base calling, or copy number.

Another widely used tool is REAPR (Recognition of Errors in Assemblies using Paired Reads) (Hunt et al. 2013). REAPR is a tool developed by the Wellcome Trust Sanger Institute, proposed to describe misassemblies (base errors and scaffolding errors) quantitatively, mapping reads to an assembly. It reports assembly errors and warnings, and also generates a new assembly by breaking the genome when an error is called over a gap. Error regions within contigs are replaced with Ns.

Both REAPR and Feature-Response Curve (FRCurve) were present in the second edition of Assemblathon (Bradnam et al. 2013). FRCurve establishes a new curve constructed on features from the *amosvalidate* software (Phillippy et al. 2008). The curve aims to better express the quality of an assembly, primarily when comparing two or more assemblies. This strategy came from ROC (receiver-operating characteristic) curves, exploiting the relationship between the metrics used in evaluations.

Another proposed strategy was CGAL (Rahman and Pachter 2013), a software application which implements a likelihood-based approach to assembly evaluation. It evaluates the uniformity of coverage, according to errors in the reads, inserts size distribution, and unassembled data. The authors reported it as better than N50 and *amosvalidate*. Besides, they analyzed the data in both the CGAL and FRCurve, leading to similar findings.

Mauve (Darling et al. 2011) was a method proposed in contrast to amosvalidate (Phillippy et al. 2008). Its main contribution is related to new features, which are: location of miscalled bases, missing segments, and other

[4]www.orthodb.org

segments; GC content of the missing and extra regions; misassemblies identified as rearrangement breakpoints inside contigs; double cut and join (DCJ) distance (Bergeron et al. 2006); and protein-coding sequences check. Also constructed on a base level, Lo et al. formalized a problem called Coverage Sensitive many-to-many min-cost bipartite Matching (CSM), a generalization from (one-to-one) weighted bipartite matching problem (Lo et al. 2013). Concretely, they developed a java tool named SAGE (Scoring function for Assembled Genomes).

More recently, we find the U50 (Castro and Ng 2017) and HiMMe (Abante et al. 2017) strategies. U50 is a new type of N50 metric that counts unique target-specific contigs against a reference genome. And HiMMe, similar to BUSCO, is a tool that provides quality metrics for genome assembly based on genetic content through Hidden Markov Models (HMM). Although, HiMMe requires prior knowledge about the organism — since it analyzes how similar the genetic patterns found in the input are to those already known –, it does not require a reference genome. It works with sequences from closely related species and/or with data previously known, persisted in a database made up of homologous genes.

2.3 Discussion

From the myriad options arises a challenge: How are scientists supposed to choose the best metrics, *i.e.,* the ones which better explain assembly quality? Vezzi et al. (Vezzi et al. 2012) analyzed amosvalidate (Phillippy et al. 2008) features and metrics such as N50 and number of contigs, applying Principal Component Analysis (PCA) and Independent Component Analysis (ICA). As a result, they found the famous N50 metric weak — "high N50 values are simply a consequence of misassemblies and due to the fact that many assemblers try aggressively to merge as many sequences/sub-contigs as possible" (Vezzi et al. 2012, p.6) — and most features redundant.

Phillippy et al. had already noticed that contiguity features exhibit high sensitivity (higher than 98%), but low specificity (Phillippy et al. 2008). Furthermore, the authors reduced the Feature-space through ICA to generate a new FRCurve.

Since amosvalidate (Phillippy et al. 2008), analyzed in (Vezzi et al. 2012), some other metrics have been proposed. In most cases, we can group them into three major categories: Base errors, Contiguity and Genomic Analysis.

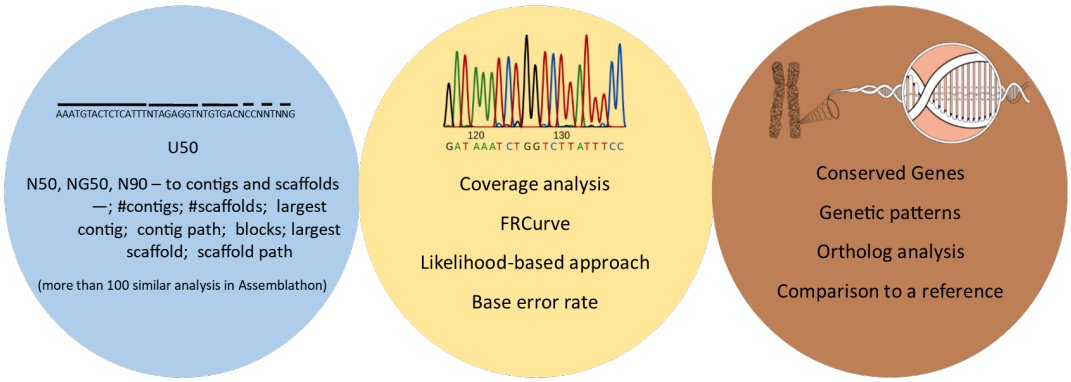

Figure 2.1: Main Assembly Evaluation Strategies

2.3.1 Contiguity Analysis

The first characteristic analyzed, when an assembly is accomplished, is its contiguity. The more contiguous and less broken an assembly is (*i.e.,* with fewer gaps), the better. In this group, we measure contigs size, number of contigs, largest contig, scaffolds size, number of scaffolds, largest scaffold, N50, NG50, NA50, NGA50, U50, N90, L50, etc.

However, contiguity metrics have been criticized by many scientists (Vezzi et al. 2012; Rahman and Pachter 2013; Castro and Ng 2017). The GAGE (Genome Assembly Gold Standard Evaluations) project (Salzberg et al. 2012) reported a considerable variation in contiguity between assemblers, and no correlation with correctness metrics (base errors). When we force to connect contigs, we risk producing artificially large N50 values, amplifying misassemblies (Salzberg et al. 2012), and consequently decreasing correctness.

2.3.2 Base Errors Analysis

In a nutshell, *Base Errors* are those which "change" the right base during the assembly process (e.g., an A instead of a C). Amosvalidate (Phillippy et al. 2008), Mauve (Darling et al. 2011), CGAL (Rahman and Pachter 2013), REAPR (Hunt et al. 2013), and SAGE (Lo et al. 2013) are tools that focus on pinpointing this kind of error.

(Phillippy et al. 2008) proposed *amosvalidate* at the beginning of NGS (Hunt et al. 2013), and then many others have innovated in the field. REAPR (Recognition of Errors in Assemblies using Paired Reads) has been developed to achieve better accuracy scoring assembly positions and identify misassem-

blies without a reference genome (Hunt et al. 2013). It has become one of the most common algorithms to assess an assembly. It outputs "error-free" and "error" bases for each position of a sequence and generates a new assembly broken at the error points.

2.3.3 Genomic Analysis

At this point, we compare genes. One approach is to verify an assembly by aligning it against a reference genome. But in the case no reference is known, we can still analyze ortholog genes, as done in BUSCO (Simão et al. 2015) or HiMMe (Abante et al. 2017).

BUSCO has been recently updated (Waterhouse et al. 2017), and now covers data for prokaryotes, vertebrates, arthropods, fungi, nematodes, protists, and plants. After concluding the identification of orthologs, BUSCO quantitatively reports complete, duplicated, fragmented, and missing genes. HiMMe scores each contig and the whole assembly, and also provides a HiMMe coefficient for the entire genome assembly. QUAST (Gurevich et al. 2013) also seeks to predict the number of genes, through GeneMark.hmm (Lukashin and Borodovsky 1998) or GlimmerHMM (Majoros et al. 2004).

The process of genes analysis may be extended in Genome Annotation. Undoubtedly, the quality of genes is related to contiguity and base correctness and can be confirmed through transcriptome data and wet lab experimentation.

2.4 Final Considerations

The Genomics field is still somewhat recent and presents a large number of practical issues to tackle. Assembly quality is one of these urgent issues that have emerged, particularly in personalized medicine scenarios. At this point, many metrics and strategies have been proposed in order to evaluate an assembly.

Taking into account the proposed evaluation metrics, we now face the problem of choosing those that give us a better understanding of the assembly quality. In many contexts, this choice will directly influence assemblies selection and consequently genome assembly application cases. In a clinical environment, for example, assembly selection may affect the number of identified mutations of medical importance.

For that reason, it is essential to combine strategies from distinct categories, correctness (base errors), contiguity, and genomic analysis, expanding the knowledge we have about the genome and the organism under investigation. The next chapter will expose our solution to understanding the correlation and the applicability of some metrics.

3
Genome Assembly Analysis Framework

As exposed in Chapter 2, given hundreds of quality features, we face the problem of interpreting those features, in order to classify the quality of assemblies. Since our primary motivation question (*Which metrics are the most relevant when evaluating an assembly?*) does not have an easy and simple answer, we propose a generalist environment where the quality metrics may be analyzed for creating a better comprehension of the quality feature space. We describe now a domain Framework for the Genome Assembly Quality Analysis.

The main goal is to permit features comparison, considering distinct sequencing, assembling, and genomic variables. We aim to detect the sequencing and assembly parameters, and the genomic characteristics that affect the quality assembly metrics. Consequently, we could contribute to user selection of the features which better respond to real-world assembling conditions.

Firstly, a brief introduction of Framework conceptions is given, followed by the requirements of our proposed analysis application. We describe then, the framework architecture and its class and sequence diagrams.

From now on, we call the metrics/measurements by features, such as in Machine Learning. A Feature is a measurable property or characteristic that describes an object or phenomenon under observation (Christopher M. Bishop 2006). For that reason, we may say that this Framework is responsible for a Feature Analysis Process of Genome Assemblies.

3.1
Framework Concept

Generally, we can define a framework based on four main requirements: code reuse, abstraction, inversion control, and extensibility (Landin and Niklasson 1995). Here, we will use and attempt to object-oriented frameworks. According to Landin and Nikalasson, "an object-oriented framework is intended to capture the functionality common to several similar applications" (Landin and Niklasson 1995). In other words, if one identifies a certain number of common requirements between independent applications, a framework would be proposed in order to provide common functions, applicable to all applications.

The requirements of a framework include the common requirements from all applications; and the adaptability for the particular requirements. Since our research question is very embracing, it could be by itself considered a framework, including a lot of sub-applications. However, it also emerges totally independent use cases that could be benefited from the framework. One of them is the assembler choice, which refers to the decision-make of the best assembler to use. The next section describes those framework requirements.

In addition, the requirements are consequently translated into frozen spots and hot spots. Frozen spots compose the core of the framework, immutable code common to all applications. Contrarily, the hot spots are the changeable modules, the ones that may be absent, and the new ones that may be often added to the application (Fayad and Schmidt 1997; Markiewicz and Lucena 2000)

3.1.1 Requirements

In this section we describe the functional requisites of the framework. Our Framework is a domain framework for the quality analysis of the Genome Assembly, and must:

(1) handle as many types of assemblers as possible.

(2) work with multiple sequencing technology types

(3) test distinct genomic characteristics

(4) test distinct assembling parameters

(5) generate multiple features

(6) be able to add or remove any of the assemblers, features, or read generators

(7) be able to work with external raw reads or to generate its own reads

(8) be able to test hypotheses concerning the whole experiment

Any other requisites may be part of the application, and work as hot spots.

3.1.2 Architecture

We introduce the proposed architecture of GAAF as follows. The blue color refers to frozen spots and the red color to hot spots. The architecture is divided into layers, one for communication, one for control, and another for service. The components of each layer are independents and may be needed or not depending on each specific application.

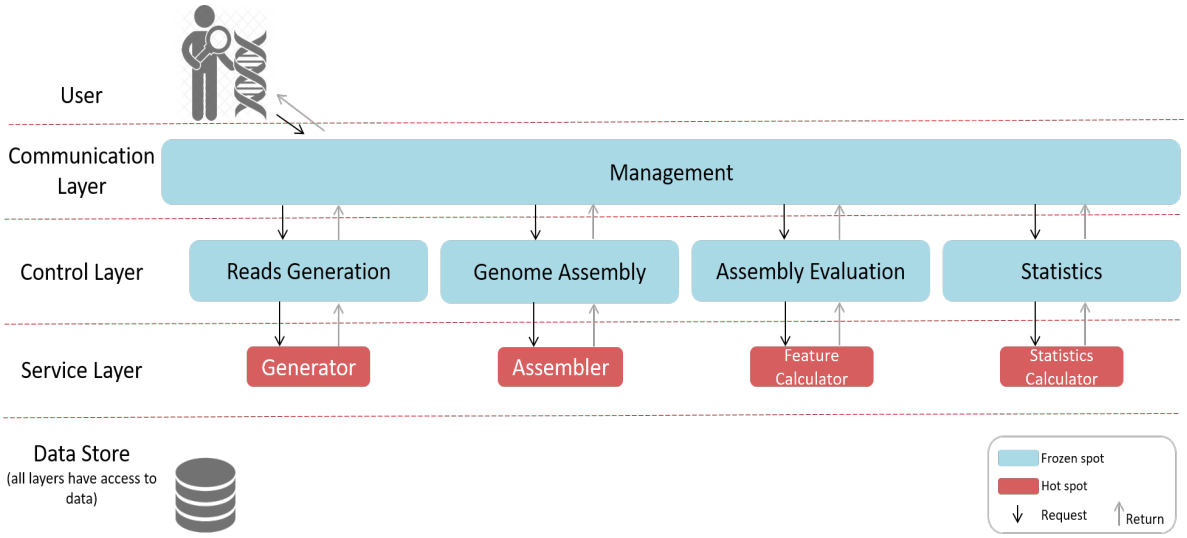

Figure 3.1: Framework Architecture

Management: all configuration, inputs, and outputs are managed through the main core module called Management. It works as the main line of communication between all other frozen spots, and between User and GAAF. Such as a Manager in an Enterprise, it acts as a communicator, passing orders to Control Layer.

Reads Generation: as the name mentions, Reads Generation Module is responsible for generating artificial reads. The way it is done depends on the chosen algorithm, e.g. pIRS (Hu et al. 2012) or Grinder (Angly et al. 2012) - in Escalona article (Escalona et al. 2016) many options are listed. The Module may receive or not a genome reference entry. When no input is available, the reads are randomly generated, according to each algorithm. In addition, it could also receive raw reads, in order to filter them in a quality trimming hot spot, or by modifying them for any purpose. Reads Generation, as well as all the other modules from Control Layer, may be seen as a Team Leader, who receives orders from Manager and passes those to specific employees.

Genome Assembly: this is the module where the reads are assembled. It may output contigs, or scaffolds, which may be analyzed in the Assembly Evaluation Module, or may work as re-input to Genome Assembly, calling hot spots capable of scaffolding, gap-filling, etc. It works with distinct assemblers, file formats, and sequencing technologies.

Assembly Evaluation: the features qualifying the assemblies are generated in the Assembly Evaluation Module. Not only the evaluation metrics, but also some other post-analysis could be included here, such as genome comparisons. Those results, according to each application, may be still analyzed in the Statistics Module.

Statistics: as a Team Leader from a group of statisticians, the Statistics Module organizes feature data and requests its service modules to test and create graphs of that data.

Generator: an abstraction of a service from Reads Generation Module is the Generator Module. Hypothetically, a freelancer, according to Client's request, who is hired to generate reads.

Assembler: another service requested is assembling. The Assembler Module is the most important service we have, it abstracts assembler software. It varies in the way of working, how outputs and how receive input data.

Feature Calculator: a good Calculator Module is needed to calculate the features. Feature Calculator may be a single class capable of calculating a single feature or may be a complex module which calculates many features at once, such as QUAST.

Statistics Calculator: finally we have a module where all statistical tests and graphs are calculated and created, respectively. The Statistics Calculator may be extended to a large variety of statistics.

The next section translates our architecture into classes, implemented in Python. Further, we explain its flow through a sequence diagram.

3.2 Class Diagram

The five frozen spots modules are presented as concrete classes. The hot spots are represented through abstract classes, which will be extended into personalized concrete classes. Figure 3-3 presents the Class Diagram, that directly reflects the architecture. Below we briefly describe the classes, please see GAAF's Manual and GAAF's code documentation for more details.

Manager: deals with the communication between user and classes, and between the classes. The Class includes the methods to generate reads, assemble reads, evaluate assemblies, load features into dataframes, and analyze those dataframes through statistical tests.

Reads_Generation_Controller: calls algorithms to generate artificial reads or to filter raw reads, given or not a reference genome input.

Generator: is an abstract class to read-generation tools. It reads a dictionary containing all the needed parameters, such as Coverage, read length, reference genome, phred score, mutation rate, sequencing technology and others. During the process of generation, the class store the sample-read names into a list.

Assembling_Controller: is responsible for managing all the assemblies required, given the chosen assemblers. It basically receives the reads, the

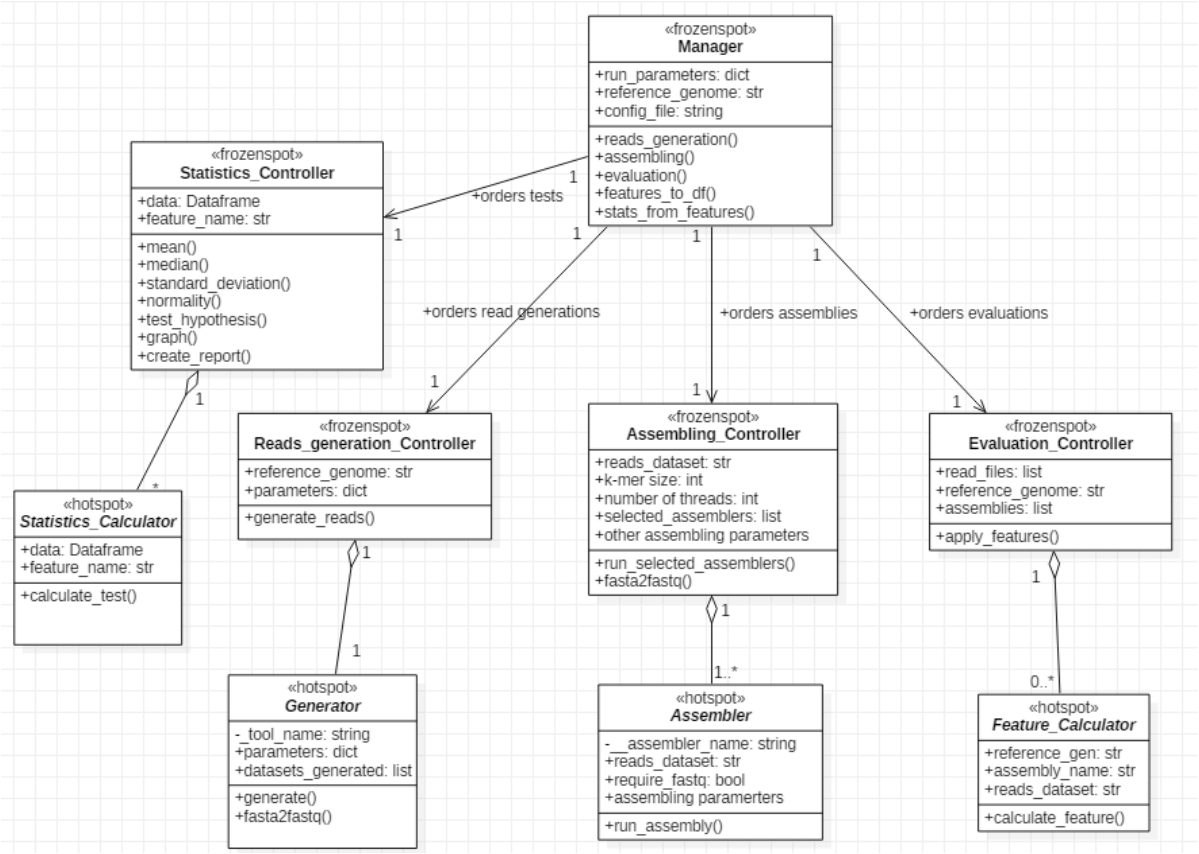

Figure 3.2: GAAF Class Diagram

k-mer size, and the number of threads, and runs the software.

Assembler: is an abstract class to the assembler software. The way each software works is implemented in a concrete class extending the Assembler Class. And it shall be invisible to the Assembling_Controller.

Evaluation_Controller: calls algorithms to evaluate the assemblies. It may call complete tools, such as QUAST, or calculate individual implemented Features.

Feature_Calculator: is an abstract class to calculate specific metrics over the assembly. It receives an assembly, and can receive a reference genome and read files.

Statistics_Controller: is another concrete class responsible for analyzing whole experiments. It has some methods already implemented in its core, such as mean and median, but generally calls Statistics_Calculator to calculate certain statistical tests.

Statistics_Calculator: is an abstract class to implement statistical tests or graphs.

3.3 Sequence Diagram

Given our architecture diagram and class diagram, we now describe a basic flow throughout GAAF. Sequence Diagram is shown in Figure 3.4.

In our implementation, an executable file, called gaaf.py (user representation), receives a config file (Figure 3.3) and some run-parameters and pass them to a Manager object. Gaaf.py request artificial reads to Manager (`reads_generation()`), and the Manager to `Reads_Generation_Controller` (`generate_reads()`). Read datasets are generated then, by an extended Generator Class, for each parameter required. The datasets are returned and stored in a predefined directory.

```
/*This is a configuration file to use GAAF. Please, change
default properties below according to your desire.
Do not leave any parameters in blank.
The chosen variable must include the values separated by
comma*/

--> Experiment name: Ecoli1

#Possible Variables: Reads len, Coverage, Phred, Mutation,
Duplication

--> Variable: Reads len
--> Coverage: 30
--> (2x)Reads len (bp):
50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,21
0,220,230,240,250,260,270,280,290,300
--> Reads len variation (Y/N): N
     --> If variate, how many bases: 0
#It considers a normal distribution.

--> Technology (Sanger, Illumina or IonTorrent): Illumina
--> Phred: 40
--> Mutation ratio: 0
--> Gene Duplication ratio: 0
```

Figure 3.3: An example of config file input to gaaf.py

The second step is asking Manager to assemble reads (`assembling()`). Manager now send the reads and a user list of selected assemblers to `Assembling_Controller` (`run_selected_assemblers()`). `Assembling_Controller` runs all selected assemblers, returning the final assemblies. They are also persisted in predefined directories.

Since user has the assemblies, one may request Manager to evaluate them (`evaluation()`), applying the features (`apply_features()`) in `Evaluation_Controller` Class. Features are calculated in one or more extended `Feature_Calculator` Classes, and stored in a directory.

Finally, the Manager converts feature files to dataframes and send them to `Statistics_Controller` (`test_hypothesis()`), where they are organized

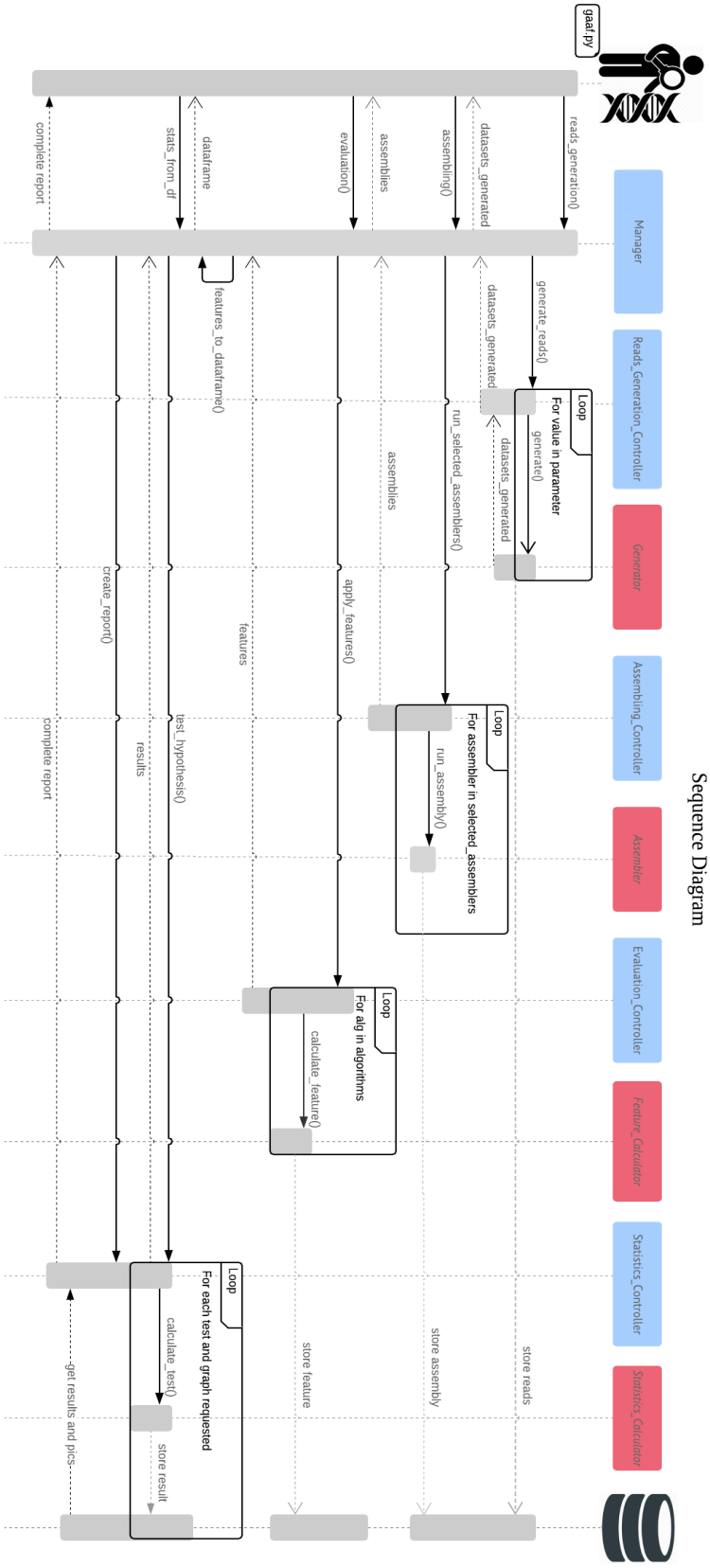

Figure 3.4: GAAF Sequence Diagram

and statistically tested in `Statistics_Calculators (calculate_test())`. At the end, user receives a complete report containing all statistical results.

When some error occurs, and/or GAAF is interrupted, a log file permits that future runs continue the process from where it was, without running everything again. In addition, it also supports reanalysis, adding new data to experiments, through new extended tools.

3.4 Extending Abstract Classes

The greatest advantage of using a framework is the easy way in how new items are added. For example, given Figure 3.5, any gray box may be removed, and any new gray box may be added.

We provide pIRS code as an example. The class is created without `__init__()` method, receiving `Generator`, from `reads_generation` module (line 5), as a Parameter (line 7). Then, `generate()` method is defined (line 29), doing whatever it needs to do. Extra methods may be created:

```
1  import os
2  import logging
3  import threading as thr
4  import time
5  from reads_generation import Generator
6
7  class Pirs(Generator):
8      """Class responsable for calling pIRS and generating its reads
9
10     Attributes
11     ----------
12     exp : str
13         The Experiment Name
14     out : str
15         The output directory to store the results and where the
       reads are stored
16     parameters : dict
17         A dictionary containing all the generation parameters
18     datasets_generated : list
19         The list of the samples generated
20
21     Methods
22     -------
23     fasta2fastq(file,ql,name='')
24         Converts a fasta file to a fastq file
25     generate(sample)
26         Run the Pirs command to the sample
27     """
```

```
28
29      def generate(self,sample):
30          """
31          Run pIRS command.
32
33          Parameters
34          ----------
35          sample : str
36              Sample name. It names the read files.
37          """
38
39          try:
40              DO WHAT IS REQUIRED
41              self.datasets_generated.append(sample)
42
43          except IOError:
44              logging.error(IOError)
```

Beside the methods created, Pirs already has `fasta2fastq()` method, and `exp`, `out`, `datasets_generated`, a dictionary of `parameters`, and `logging` attributes. `Exp` gives the name of the experiment and `out` the output directory to store the reads. It is recommended to create "/reads" dir inside `out`. It is important to save the sample names created into a predefined list called `datasets_generated`. It will be used further by other modules.

Finally, the `parameters` dictionary gives the class all experiment parameters passed by user. Not all of them may be used. It is up to developer. At least, one will probably use the following keys: `ref`, `coverage`, `read_len`, `var` (meant read normal variation), `phred` and `Error_rate`.

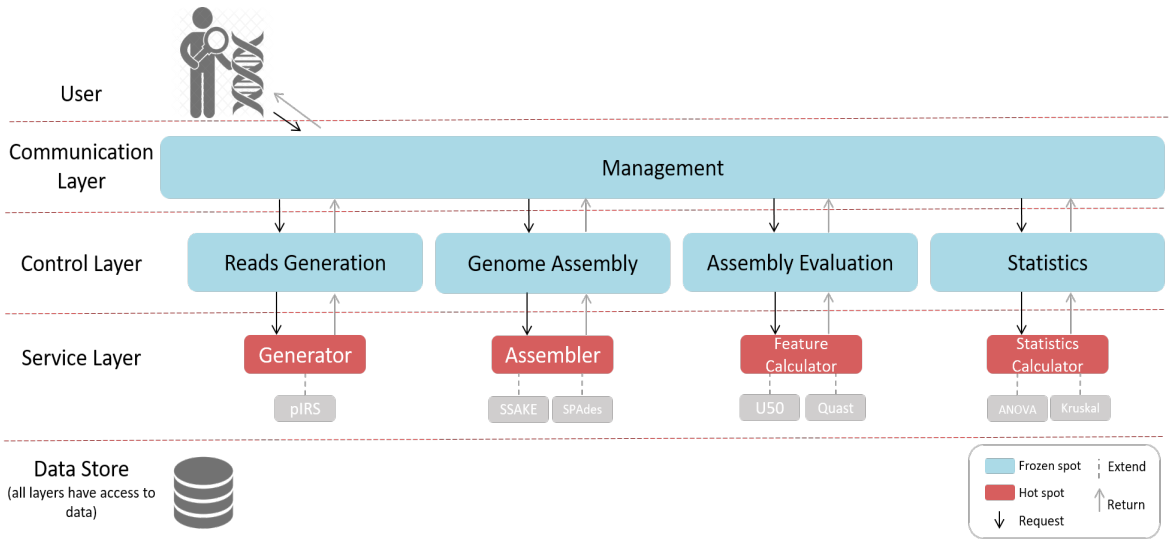

Figure 3.5: GAAF Architecture with some Hotspots Instantiated

The Framework Manual, provided on Github and in the Supplementary Material, explains that procedure to each abstract class. It is always the same: receive the abstract class as a parameter and define the abstract method and extra methods required by specific applications.

3.5 Unit Tests

During implementation, some unit tests were performed to testify classes operation. The results can be accessed in `unit_tests/tests` (https://github.com/neumannguib/GAAF-Framework).

The next chapter presents a Study Case, our motivation application, to test and validate our Framework.

4
Case Study

Considering our primary motivation question (*Which metrics are the most relevant when evaluating an assembly?*), and a second auxiliar motivation question (*What do the features represent in the genome assembly?*), we proposed GAAF, as exposed in Chapter 3. The layers and the modules presented are trying to provide the main steps needed in experiments that would try to answer these motivation questions.

However, in order to answer them to the whole domain, including all living beings, it is necessary to answer before the same questions to sub-domains, or sub-groups. We decided then, to first respond that to a single bacterium – *Escherichia coli*.

As a case study, to test our Framework, we analyzed the correlation between the Genome Assembly Evaluation Features and the following sequencing parameters: read length, read coverage and phred quality. Moreover, we compared four strains of *Escherichia coli*. The main goal was to detect the influence of sequencing-parameters/genomic characteristics over the assembly features. Hereafter, after analyzing a bigger Feature Space, and a wider range of parameters, it will be possible to infer, from the features, which parameters and/or genomic characteristics are varying in the assemblies of *Escherichia coli*.

Firstly, *Escherichia coli* was chosen as our genomic model, since *E. coli* naturally presents high intraspecific diversity. It has non-pathogenic and pathogenic strains, living in several different environments, composed of some distinct genomic configurations (Moriel et al. 2012) - see Table 4.1. The strain k12 substrain MG1655 is the main reference genome for *E. coli* in terms of genetic studies, well described in the literature. For that reason, we used *Escherichia coli* strain k12 substrain MG1655 as our genome reference in the first four experiments.

E. coli is highly repetitive, and a first look at that in 2000 (Gur-Arie et al. 2000) had already detected thousands of repeats. So the fifth experiment will consider the natural repeat variation between *E. coli* strains to evaluate how it affects assembly quality. Four strains were selected based on repetitive regions, as described in Table 4.1. Those repetitions were analyzed by DACCOR software (Seitz et al. 2018).

Table 4.1: Strains of *Escherichia coli* and their repetitive patterns analyzed by DACCOR (Seitz et al. 2018)

	k12 MG1655	**IAI39**	**O83:H1 str. NRG 857C**	**O104:H4 str. 2011C-3493**
Accession	NC_000913.3	NC_011750.1	NC_017634.1	NC_018658.1
Genome Length (bp)	4,641,652	5,132,068	4,894,879	5,437,407
GC (%)	50.8	50.6	50.7	50.7
Genes	4,566	5,092	4,532	5,081
tRNAs	86	88	84	94
rRNAs	22	22	22	22
Pseudogenes	147	160	1	2
Proteins	4,242	4,725	4,425	4,963
Repetitive Regions	622	1,005	365	1,052
Different Repetitive Regions	248	361	168	506
Maximum Repetition length (bp)	2,815	5,230	2,770	6,581

Pursuing the motivation question, many parameters and genomic characteristics may emerge. Read Length was selected since it is one of the first parameters to ponder when sequencing. The technology and sequencing kit one uses, defines read length. In terms of Illumina sequencing, reads currently vary between 150bp and 300bp in length (Illumina 2019).

Secondly, coverage and phred quality are related to the sequencing purpose. On the one hand, coverage can be understood as the number of times a position in the genome is represented in the pool of reads - reads that overlap that position. Coverage 10x, for example, means that each position in the assembly appears at least in 10 reads (Sims et al. 2014). Read coverage requirement depends on the application, for detecting human genome mutations, SNPs, and rearrangements, for example, Illumina indicates coverage between 10x to 30x, while 100x for ChIP-Seq (Illumina 2014).

On the other hand, phred says how many errors are implicitly present in the Genome. It directly points to the probability of correct base detection in the sequencing process. Normally, it is expected a phred greater than Q30 (Illumina 2011). This is equivalent to the probability of an incorrect base call 1 in 1000 times (Table 4-2). In scenarios where a single base is important, these

two aspects of coverage and phred are crucial.

Table 4.2: Quality Scores and Base Calling Accuracy modified from (Illumina 2011)

Phred Score	**Probability of Incorrect Base Call**	**Base Call Accuracy**
Q10	1 in 10	90%
Q20	1 in 100	99%
Q30	1 in 1,000	99.9%
Q40	1 in 10,000	99.99%
Q50	1 in 100,000	99.999%

Other parameters such as the Mutation rate, indels rate, k-mer size, Duplication rate, Genomic Complexity etc., shall be analyzed further; not only to *E. coli*, but also to other species and taxonomic groups.

We generated artificial Illumina reads, according to desirable characteristics (see Table 4.3), using pIRS v2.0.0 (Hu et al. 2012). pIRS was selected based on Escalona article, experiment requirements, and our requirements for recent releases and available support (Escalona et al. 2016). For this case study, we only generated Illumina reads due to fact that other sequencing technologies, as discussed in Chapter 1, that generate longer reads, still have high substitution error rate, around Q10, and are normally used in conjunction with Illumina reads. For that reason, we decided to first analyze Illumina reads, independently, and in future studies, use the other technologies too. Except for the parameter in analysis (generated in pIRS, or present in the reference genome), all other parameters are fixed.

All experiments run 8 distinct assemblers: SPAdes v3.13.0 (Bankevich et al. 2012), MIRA V5rc1 (Chevreux et al. 2004), Velvet v1.2.10 (Zerbino 2010), ABySS v2.0 (Jackman et al. 2017), Minia v3.2.0 (Chikhi and Medvedev 2013), SSAKE v4.0 (Warren et al. 2007), Edena v3.131028 (Hernandez et al. 2008) and Masurca v3.3.1 (Zimin et al. 2013). These assemblers vary in algorithm strategies (Table 4.4), are commonly used, and were present in GAGE-B work (Magoc et al. 2013) and in Assemblathon 1 (Earl et al. 2011). The assemblers choice was also based on our requirements for recent releases and available support.

For the k-mer-based assemblers, even for those which use internal methods to discover the best k size, KmerGenie v1.7051 (Chikhi and Medvedev 2013) was applied to the generation of K size. The goal was not to definitely find the best k, but just to use the same k parameter for all assemblers.

Table 4.3: Experiment Design for Strains of *Escherichia coli*

Experiments / Parameters	**E1**	**E2**	**E3**	**E4**	**E5**
Strain	K12 MG1655				K12 MG1655, IAI39, 083:H1 NRG857C, O104:H4 2011C3493
Read Length (bp)	50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300	30-170 40-180 50-190 60-200 70-210 80-220 90-230 100-240 110-250 120-260 130-270 140-280 150-290 160-300 170-310 180-320	250	100	250
Coverage	30x		10x 30x 50x 70x 90x 110x 130x 150x		30x
Phred	Q40			Q1 Q5 Q10 Q15 Q20 Q25 Q30 Q35 Q40	Q40

We calculated the features using QUAST v5.0.2 (Gurevich et al. 2013), from Contigs, generating Latex files. From these files, 18 features, from the three groups presented in the second chapter, were selected: Reference mapped (%), Total length, N50, NA50, NG50, L50, N75, Number of Contigs, Largest Contig, GC Content (%), Mapped (%), Number of mismatches per 100 kbp, Number of misassemblies, Largest alignment, Average Coverage Depth, Number of Predicted Genes (unique), Complete BUSCO (%), and Partial BUSCO (%). All these selected features are described in Appendix A, according to QUAST (Gurevich et al. 2013).

We analyzed these 18 features for each experiment through a Multisample Analysis of Variances such as ANOVA (Kaufmann and Schering 2014) or Kruskal-Wallis (Kruskal and Wallis 1952), per parameter, and also per assembler - see below our hypothesis. When the null hypothesis was denied, Nemenyi test (Nemenyi 1963) or Bonferroni test (Dunnett 1955) was applied to the results. Shapiro–Wilk (Shapiro and Wilk 1965) performed Normality tests, indicating the use of ANOVA or Kruskal-Wallis tests.

When any statistical difference was detected, Correlation analysis of Pearson was applied, and Linear Regression to results when correlation was above 0.80.

Related to Parameters:

H0: There is no significant difference in Feature X between the values from parameter Y.

H': The values from parameter Y have some influence over Feature X.

Related to Assemblers:

H0: There is no significant difference in Feature X between the assemblers.

H': The assemblers have some influence over Feature X

We installed and run GAAF in Ubuntu 18.04.2 LTS System, 32GB RAM, Intel Xeon E312xx 16 Cores machine.

4.1 Results

In this section, we present the main results for all experiments. The complete reports are available in the Supplementary Material. The analysis presented here are the ones applied to the features after the process of outliers removing. The outliers were removed using Z-score test ($z>|3|$).

Table 4.4: Assembly Strategies

Assembler	**Strategy**	**K-mer Use**
SSAKE	Greedy	yes
SPAdes	de Bruijn graph	yes
MIRA	OLC	no
ABySS	de Bruijn graph	yes
MaSuRCA	hybrid (OLC + de Bruijn)	yes
Minia	de Bruijn graph	yes
Velvet	de Bruijn graph	yes
Edena	OLC	no

4.1.1 Experiment 1: Read Length

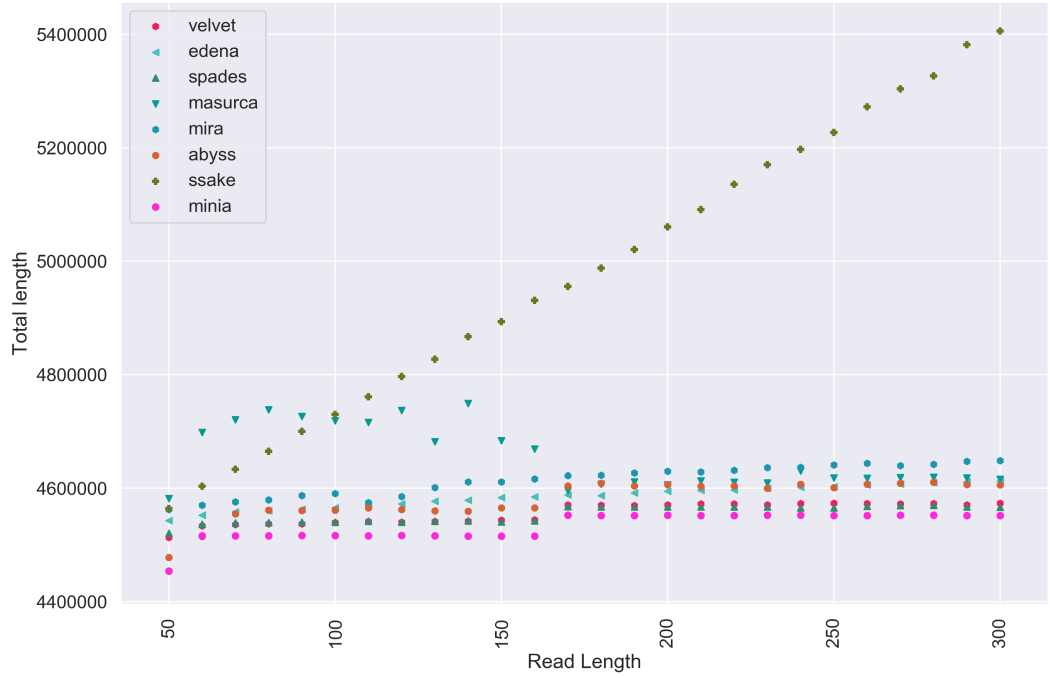

Figure 4.1: Total Length (bp) results in read length datasets from 50 to 300 bp.

The First Experiment focused on the influence of Read Lengths over the assembly features. We generated 26 read datasets from 50 bp to 300 bp, which were assembled on 8 assemblers, totaling 208 assemblies. Assemblies from SSAKE were eliminated from the analysis since it generated assemblies much larger than the reference genome - Figure 4.1. In the discussion section we

better comment that behavior. Except for SSAKE, all assemblers had assembly sizes close to the reference: 4.6Mbp.

Table 4.5: Summary Results of Experiment 1. For each Feature, we can see p-values when comparing distinct assemblers, and when comparing distinct read lengths. A p-value below 0.05 means that there is a significant difference between assemblers or read lengths at that Feature.

Feature	**p-value Assemblers**	**p-value Read Length**
Complete BUSCO (%)	0.0603	0.5122
Partial BUSCO (%)	0.1889	0.5625
GC Content (%)	**1.48e-21**	0.0933
# Predicted Genes	**6.60e-23**	0.1629
Mapped (%)	**5.52e-15**	**7.50e-07**
Avg coverage depth (%)	**4.49e-12**	0.9969
Reference Mapped (%)	1.0	1.0
Total length	**1.76e-24**	0.2344
L50	**4.54e-16**	**1.30e-06**
Largest alignment	**1.53e-17**	**0.0009**
Largest contig	**6.62e-16**	**0.0011**
N50	**7.32e-17**	**3.98e-07**
NA50	**4.95e-17**	**5.21e-07**
NG50	**6.81e-17**	**5.49e-07**
N75	**7.54e-17**	**6.36e-07**
# Contigs	**8.92e-19**	**0.0002**
# Misassemblies	**2.07e-20**	0.9743
# Mismatches per 100 kbp	**2.06e-21**	0.5315

KmerGenie (Chikhi and Medvedev 2013) was used to generate the best k sizes to apply in all assemblers. However, as read lengths grew, some k sizes needed to readjust. So the chosen sizes were as follows: 21 for Read Length 50bp, 31 for 60 to 160bp, and 81 for 170 to 300bp.

Figure 4.2: N50 (bp) results in read length datasets from 50 to 300 bp.

Almost all features presented significant statistical differences between the assemblers. Assemblers of distinct assembling strategies showed significant differences in Nemenyi test, e.g. Masurca and Minia and Edena and Minia in N50, ABySS, and MIRA in N75. However, still assemblers from the same strategy groups had some differences, such as in N50, Minia and SPAdes, and Velvet and Minia and Minia and SPAdes in N75 (see Complete Report Experiment 1 of Supplementary Material for more details). It is clear in figure 4.2, for example, how assemblers work differently, in N50 output.

Related to the read length, the features L50, Largest Alignment, Largest Contig, N50, N75, NA50, NG50, Number of Contigs, and Mapped (%), presented statistically significant differences - see Table 4-4. However, no punctual divergences were detected through Nemenyi between pairs of read lengths. It may mean that only contiguity features have been influenced by Read Length. In addition, almost all of them are correlated to Read Length (Figure 4.3).

Analyzing Linear Regression, in the cases of Pearson correlation detection, except in Largest Alignment and Largest Contig for Masurca, all assemblers exhibited significant linear regression values.

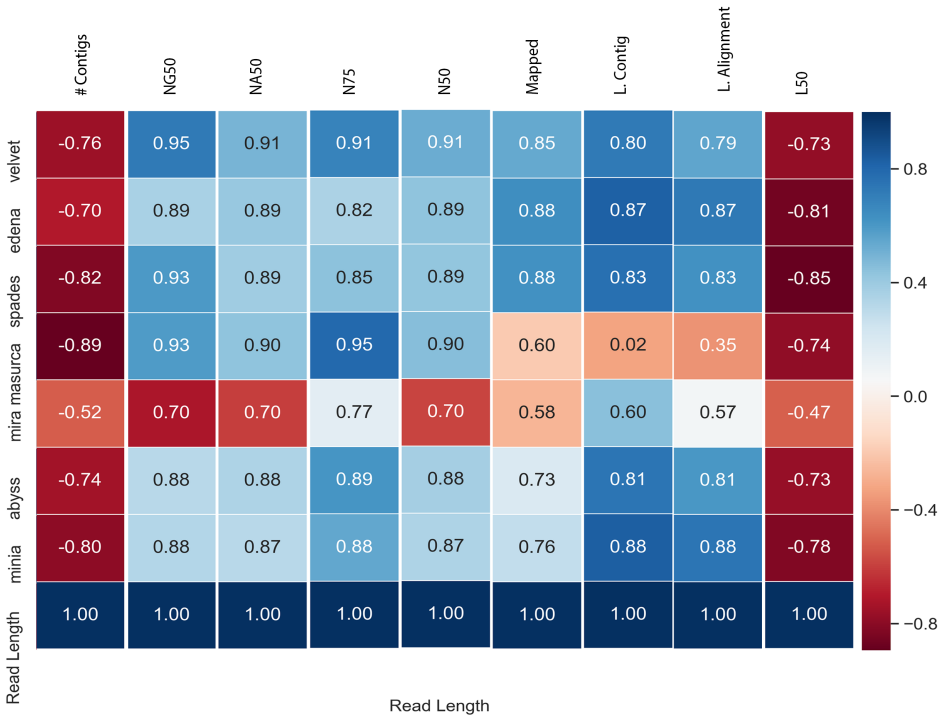

Figure 4.3: Pearson Correlation Results between Read Length and Assemblers for Contiguity Features in Experiment 1

4.1.2 Experiment 2: Normal Distribution of Read Lengths

The second experiment had the same objective as stated in experiment 1, changing just in the point where read lengths had a normal distribution, such as in real world. It means a read dataset of length 100bp, with a variation of 70bp, varies from 30bp to 170bp, but with an approximately mean length of 100bp.

Sixteen datasets were assembled on 8 assemblers, totaling 128 assemblies. SSAKE was also not considered into the analysis. K-mer sizes were 31 for reads of 100 to 160bp, and 81 for 170 to 250bp.

It is interesting to observe that in this experiment all features diverged between assemblers (Table 4-6), but fewer diverged in terms of read length, in comparison to the first experiment. Total length, number of contigs and N75 do not statistically change when reads variate.

Both experiments 1 and 2 had Mapped (%) feature significantly changing. That means larger reads better map to the assembly (Figure 4.4), as well as generate assemblies with fewer gaps; without any statistical differences to correctness (#misassemblies and #mismatches) neither to genomic analysis

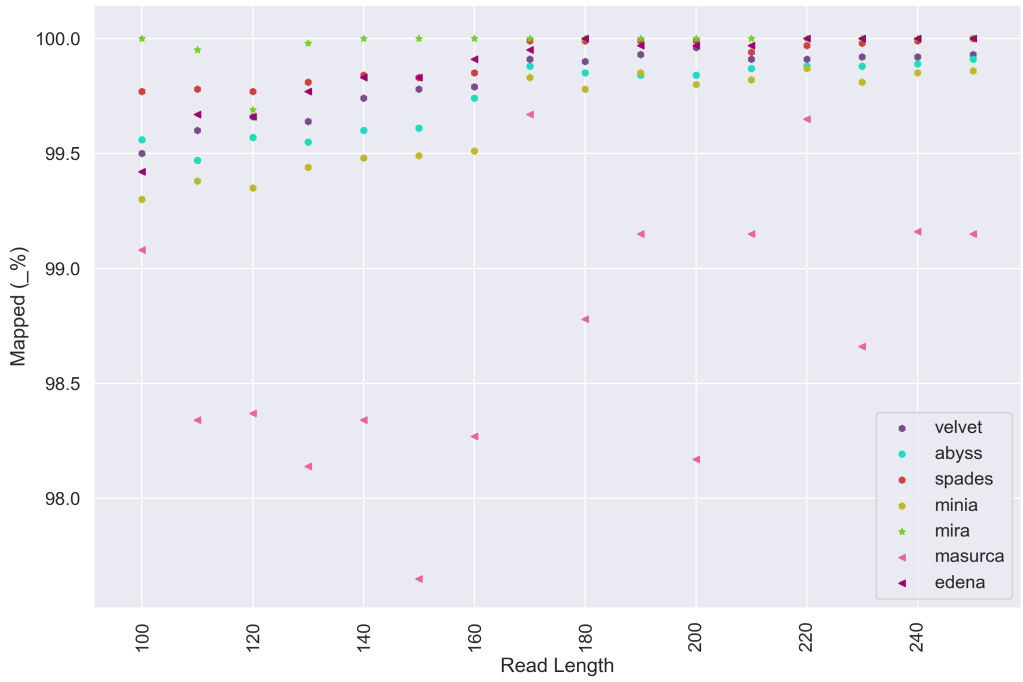

Figure 4.4: Percentage of mapped reads to the assemblies in Experiment 2

(BUSCO, predicted genes, and GC content).

Those significant features also were correlated to read length (Figure 4.5), and the ones with a correlation value greater than 0.80 demonstrated a significant regression coefficient too.

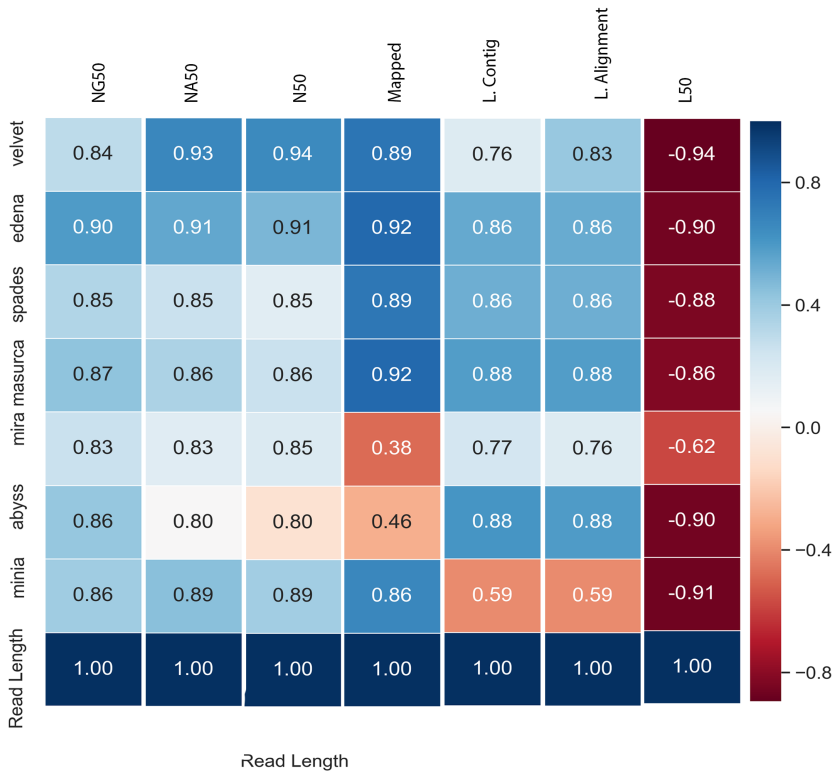

Figure 4.5: Pearson Correlation Results between Read Length and Assemblers for Contiguity Features in Experiment 2

Table 4.6: Summary Results of Experiment 2. For each Feature, we can see p-values when comparing distinct assemblers, and when comparing distinct read lengths. A p-value below 0.05 means that there is a significant difference between assemblers or read lengths at that Feature.

Feature	**p-value Assemblers**	**p-value Read Length**
Complete BUSCO (%)	**1.39e-15**	0.9997
Partial BUSCO (%)	**2.42e-05**	0.7227
GC Content (%)	**1.48e-21**	0.9756
# Predicted Genes	**1.52e-14**	0.2431
Mapped (%)	**5.04e-13**	**0.0096**
Avg coverage depth (%)	**1.89e-21**	0.9999
Reference Mapped (%)	1.0	1.0
Total length	**1.05e-17**	0.8082
L50	**1.09e-12**	**0.0010**
Largest alignment	**2.34e-05**	**4.59e-06**
Largest contig	**4.40e-05**	**1.62e-05**
N50	**1.24e-12**	**0.0011**
NA50	**1.24e-12**	**0.0012**
NG50	**2.45e-07**	**2.20e-06**
N75	**7.33e-15**	0.0900
# Contigs	**1.02e-14**	0.0719
# Misassemblies	**1.24e-14**	0.9928
# Mismatches per 100 kbp	**9.83e-14**	0.2242

4.1.3 Experiment 3: Coverage

Experiment 3 focused on how coverage parameter affects the genome quality features. Coverage varied between 10x and 150x, with 8 datasets assembled on 8 assemblers, totaling 64 assemblies. However, the results from coverage disregard SSAKE and MIRA assemblies.

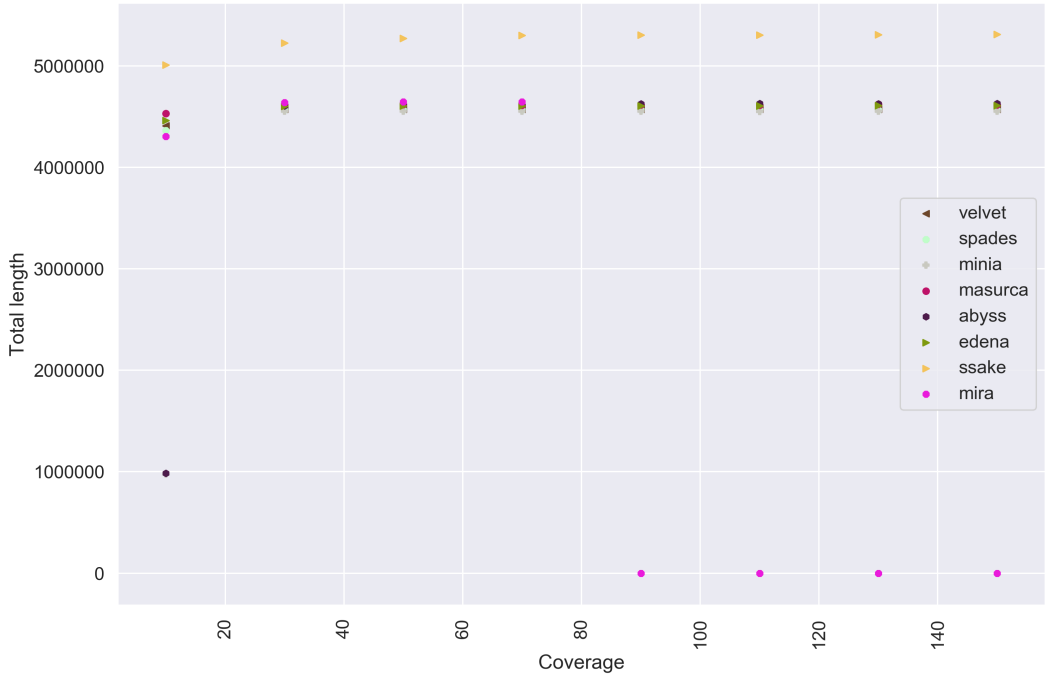

Figure 4.6: Total Length (bp) in Experiment 3

On the one hand, SSAKE was removed due to the same reason from the other experiments: poor assemblies. On the other hand, MIRA had a hard time assembling high coverage. It did not assemble reads with coverage of 90x,110x,130x or 150x (Figure 4.6). SSAKE and MIRA are the oldest assemblers in this work and were developed in the context of a bite distinct sequencing technologies. K size 17 was used for 10x, and 81 for 30 to 150x.

Such as in experiments 1 and 2, contiguity features demonstrated significant divergences between coverage parameters (through Kruskal test) - Table 4-7. However, Nemenyi tests only presented divergences in the features Complete BUSCO and Partial BUSCO, between Coverage 10x and all the others. We may conclude, then, that very low coverage affects conserved genes (below 10x), and, consequently, Complete BUSCO (Figure 4.7) and Partial BUSCO may be indicators of low coverage. Contiguity features can be affected, but they

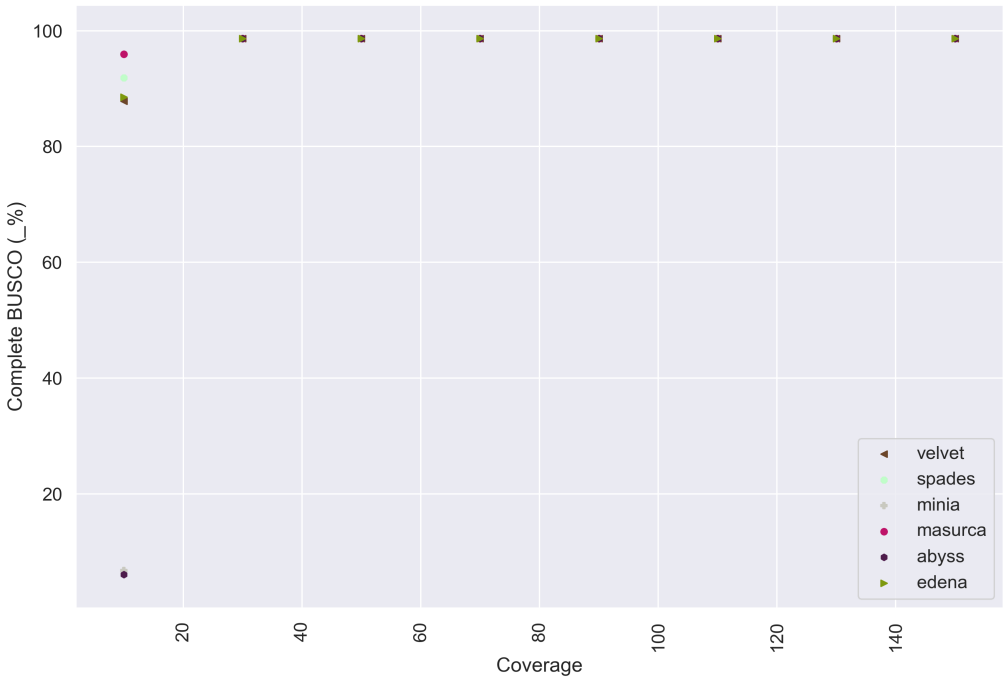

Figure 4.7: BUSCO Results in Experiment 3

will be weaker indicators of changes in coverage, since they are also indicators of Read Length.

Average coverage depth(%) feature helps us to confirm the changing in read coverage, as expected. Number of misassemblies was also significant, but it was caused yet by the high number of misassemblies in coverage 10x, as well as smaller assemblies. We need, although, to review this experiment, since k size was just different in coverage 10x, a potential bias to the presented results.

Table 4.7: Summary Results of Experiment 3. For each Feature, we can see p-values when comparing distinct assemblers, and when comparing distinct coverage values. A p-value below 0.05 means that there is a significant difference between assemblers or coverage values at that Feature.

Feature	**p-value Assemblers**	**p-value Coverage**
Complete BUSCO (%)	0.9999	**6.30e-08**
Partial BUSCO (%)	0.9999	**6.28e-08**
GC Content (%)	**1.74e-05**	0.3046
# Predicted Genes	**2.432e-06**	0.9915
Mapped (%)	**8.78e-05**	**0.03568**
Avg coverage depth (%)	0.9991	**6.54e-08**
Reference Mapped (%)	1.0	1.0
Total length	**7.435e-05**	**0.02557**
L50	**6.25e-05**	**0.0174**
Largest alignment	**7.84e-05**	**0.0233**
Largest contig	**7.84e-05**	**0.0233**
N50	**6.47e-05**	**0.0260**
NA50	**6.47e-05**	**0.0260**
NG50	**6.43e-05**	**0.0270**
N75	**5.94e-05**	**0.0257**
# Contigs	**8.91e-05**	**0.0282**
# Misassemblies	0.0955	**0.0125**
# Mismatches per 100 kbp	**1.77e-06**	0.9867

4.1.4 Experiment 4: Phred

The fourth experiment took a deeper look at phred quality. We simulated distinct phred values through the substitution error rate, as shown in Table 4.8. Approximately (error rate is a probability), though, phred varied between Q10 and Q50, on 8 assemblers, totaling 72 assemblies.

Velvet was removed from the analysis because it did not assemble reads for phred of Q5 and Q15, and generated assemblies bigger or smaller than reference genome length. Even though, many assemblies were smaller than the reference in phred below Q25 (Figure 4.8).

Table 4.8: Quality Scores and respective Substitution Error Rates

Phred Score	**Substitution Error Rate**
Q1	0.79
Q5	0.31
Q10	0.1
Q15	0.031
Q20	0.01
Q25	0.0031
Q30	0.001
Q35	0.00032
Q40	0.0001

Again, contiguity features had null hypothesis denied for the parameter change, in this case, error rate. Notwithstanding, Complete BUSCO and Partial BUSCO were also statistically divergent in phred change. And it totally makes sense, since more errors may cause gaps in gene sequences (Figures 4.9 and 4.10). Despite that, Nemenyi did not detect specific pair divergences, for BUSCO features.

The total length of assemblies was also smaller below Q25. Similarly, we had smaller contigs and N50, NA50, NG50, N75. The curious point is the fact that we could not deny the null hypothesis to Number of misassemblies neither to Number of mismatches, even with higher error probabilities.

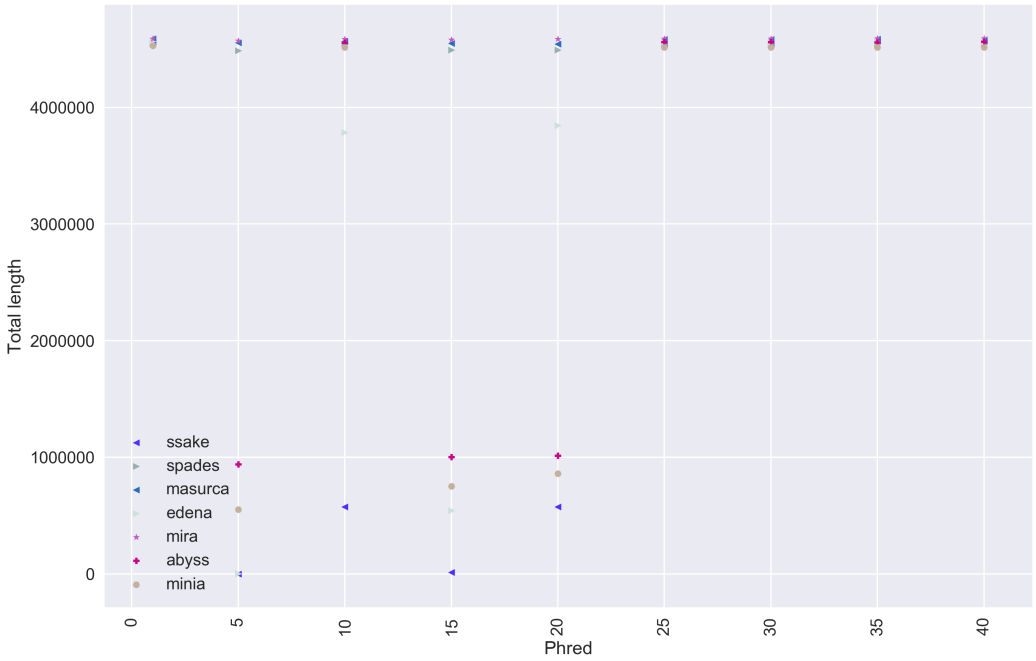

Figure 4.8: Total Length(bp) in Experiment 4

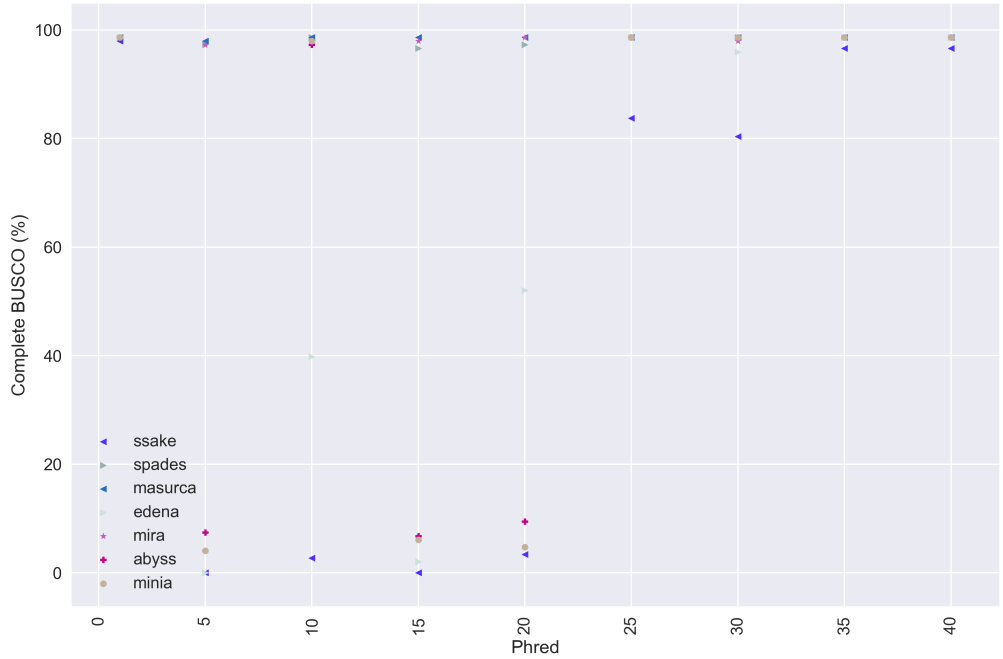

Figure 4.9: Complete BUSCO (%) in Experiment 4

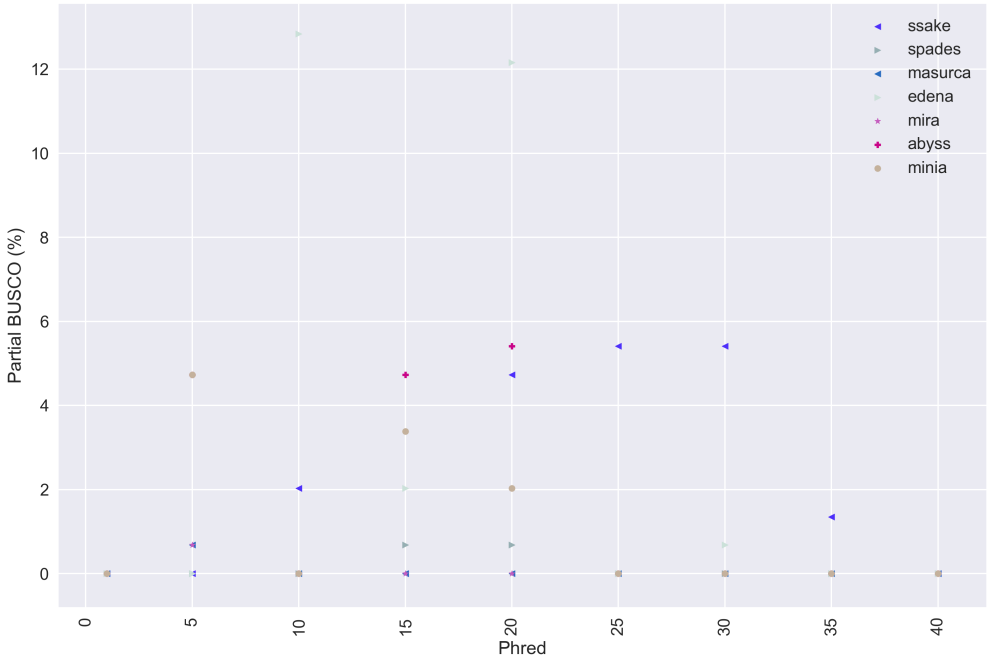

Figure 4.10: Partial BUSCO (%) in Experiment 4

Table 4.9: Summary Results of Experiment 4. For each Feature, we can see p-values when comparing distinct assemblers, and when comparing distinct phred values. A p-value below 0.05 means that there is a significant difference between assemblers or phred values at that Feature.

Feature	**p-value Assemblers**	**p-value Phred**
Complete BUSCO (%)	**0.0022**	**0.0002**
Partial BUSCO (%)	0.1686	**0.0303**
GC Content (%)	**8.56e-05**	0.8765
# Predicted Genes	**0.0023**	0.6442
Mapped (%)	**8.92e-05**	**0.0028**
Avg coverage depth (%)	**0.0056**	0.9823
Reference Mapped (%)	1.0	1.0
Total length	**4.62e-06**	**0.0140**
L50	**0.0020**	0.2619
Largest alignment	**4.52e-05**	**0.0025**
Largest contig	**4.05e-05**	**0.0022**
N50	**1.88e-05**	**0.0057**
NA50	**2.72e-05**	**0.0045**
NG50	**2.15e-05**	**0.0055**
N75	**1.07e-05**	**0.0101**
# Contigs	**0.0022**	0.2681
# Misassemblies	**5.87e-07**	0.9923
# Mismatches per 100 kbp	**3.43e-06**	0.3350

4.1.5 Experiment 5: *Escherichia coli* strains

Dislike other experiments, experiment 5 did not simulate, create nor alter any parameters of sequencing. This experiment only compared those four strains, as exposed in Table 4.1. We would think that in many species, to compare intra-specific diversity will probably not cause statistical divergences in assembly quality features. But when we talk about bacteria, and about *E. coli*, it may change.

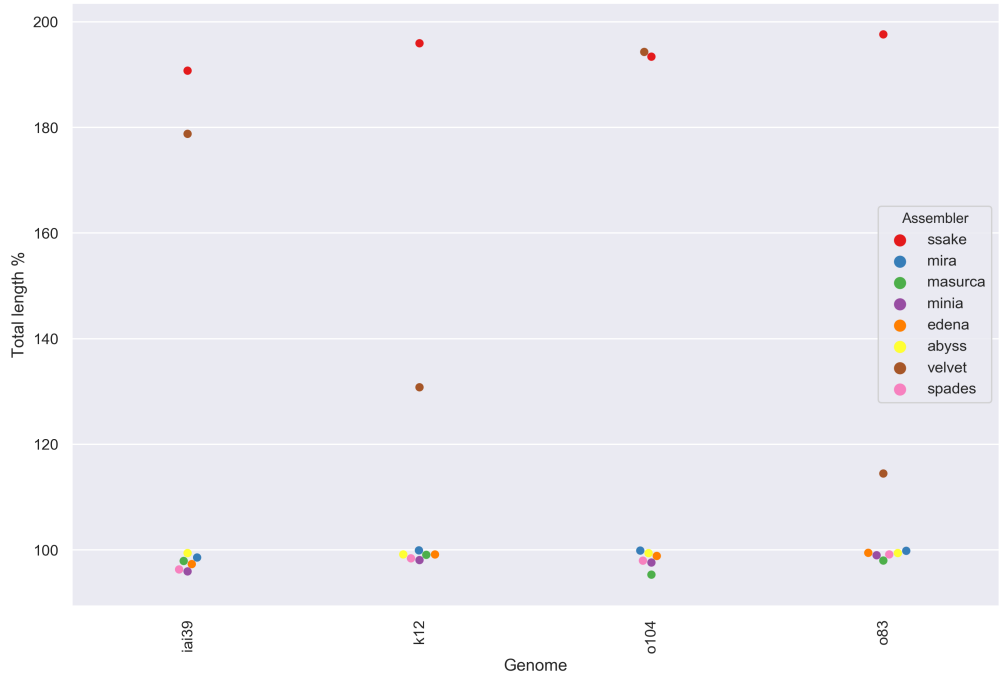

Figure 4.11: Percentage of total assembly length to strain genome length

Could N50 be affected by repetitive regions that differ in *E. coli* ? Yes, and it definitely does. Table 4-10 shows, one more time, that contiguity features statistically differ, and prove again how sensitive they are.

We had 32 assemblies. The features were normalized when necessary, according to total length, aligned length or reference genome length. SSAKE and Velvet were removed from the analysis, also by problems in total length (Figure 4.11).

We have found some interesting results relating repetitive strains to some features. As already exposed in Table 4.1, strain O104 has the highest number of repetitive regions, while O83 the least. Let us firstly focus on genomic analysis.

The number of genes is known for each strain (Table 4-1). We analyzed then, the difference between the number of predicted genes and the number of known genes; in other words, the number of unpredicted genes. There is a statistically significant difference between them (Table 4-10). O83 was the one closest to the real number of genes - Figure 4.12. Curiously, the one with the highest number of unpredicted genes was iai39, the second most repetitive strain. But of course might be other factors affecting genes prediction, since O104 was not the highest with unpredicted genes, not even the second highest. Nemenyi findings point to differences only between O83 and iai39.

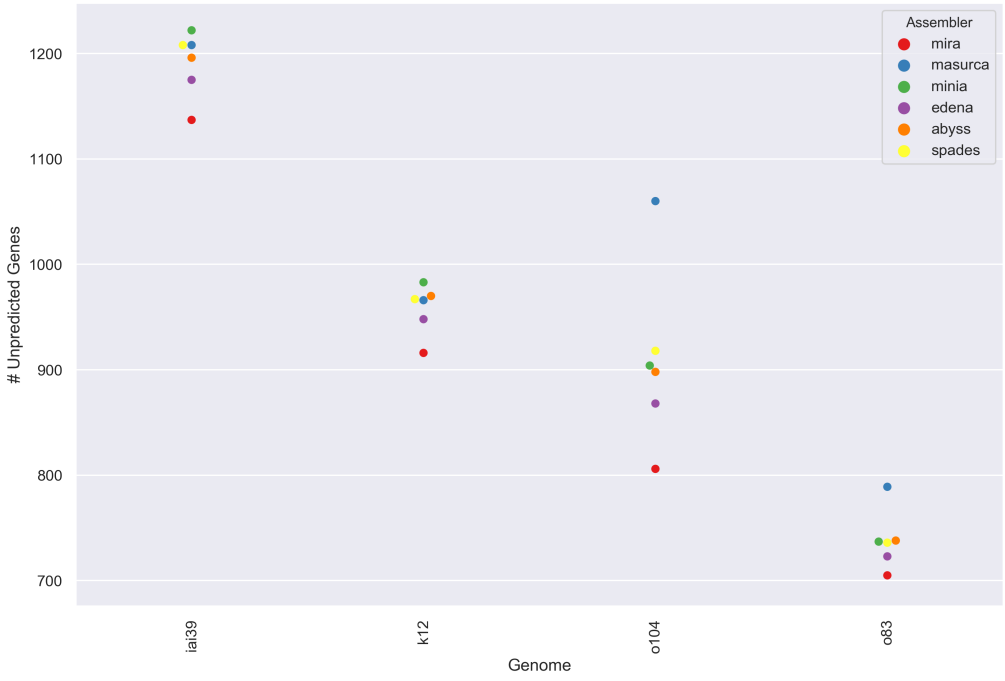

Figure 4.12: Number of unpredicted genes in Experiment 5

The same occurred in GC content. The strains are naturally quite different (Table 4-1). But when comparing the difference between the reference and the assemblies, we detected divergences within strains (Figure 4.13). Nemenyi showed that O104 is statistically different from all other strains in normalized GC content.

In contiguity features, N50 for example seemed to be very related to repetition level. O83 had an N50 largely representing the assembly, while 0104 and iai39 shortly representing (Figure 4.14), corroborated by Nemenyi findings.

While on the one hand, contiguity and genomic features are somehow related to repetitive regions in the genomes, on the other hand, base features

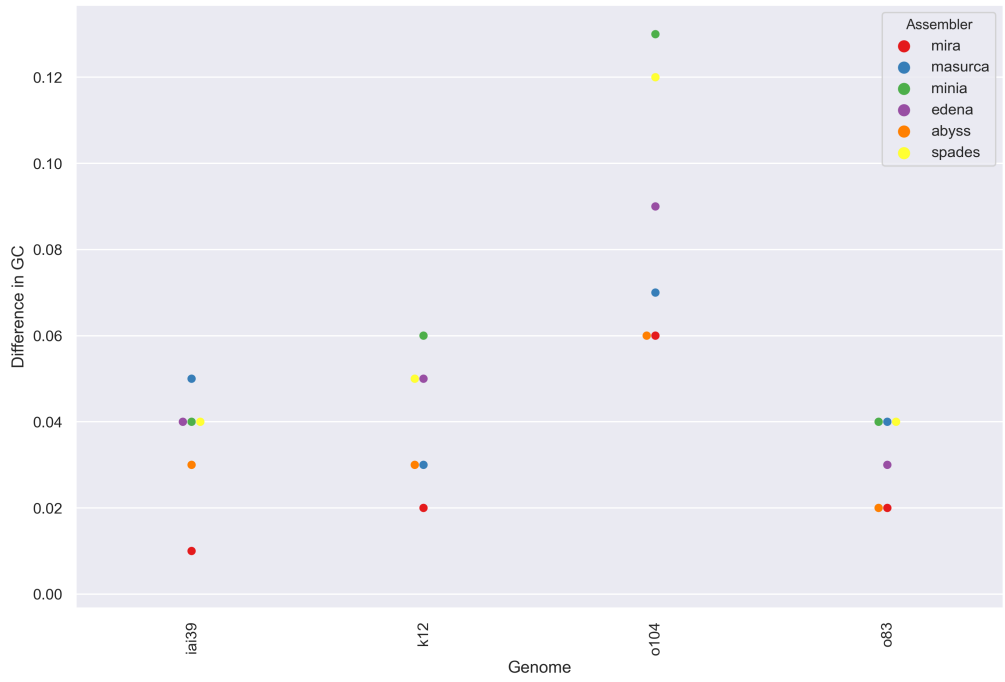

Figure 4.13: Difference in GC Content in Experiment 5

do not. Number of mismatches and misassemblies, contrarily, presented no pattern, despite significant differences between assemblies.

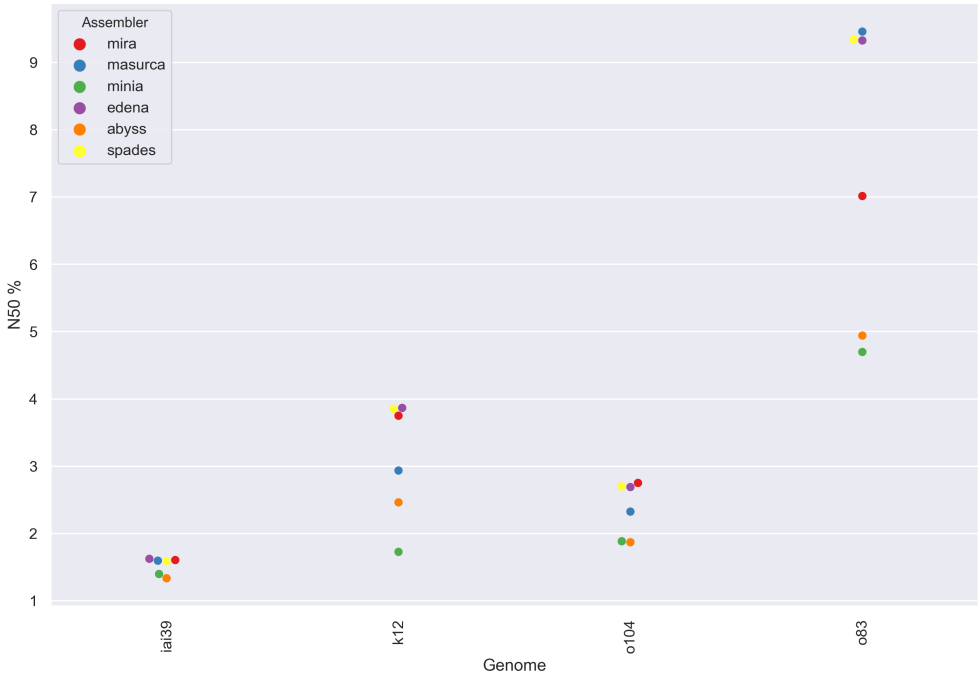

Figure 4.14: Percentage of N50 representing the assemblies in Experiment 5

Table 4.10: Summary Results of Experiment 5. For each Feature, we can see p-values when comparing distinct assemblers, and when comparing distinct E. coli strains. A p-value below 0.05 means that there is a significant difference between assemblers or E. coli strains at that Feature.

Feature	**p-value Assemblers**	**p-value Strains**
Complete BUSCO (%)	0.0638	0.5530
Partial BUSCO (%)	0.4158	0.3916
Difference in GC	0.3046	**0.0042**
# Not predicted Genes	0.8916	**0.0001**
Mapped (%)	**0.0006**	0.9905
Avg coverage depth (%)	0.5235	0.0990
Reference Mapped (%)	1.0	**4.03e-05**
Total length	**0.0294**	0.1074
L50	0.8257	**0.0001**
Largest alignment	0.9723	**0.0030**
Largest contig	0.9917	**7.90e-05**
N50	0.8393	**0.0001**
NA50	0.8914	**0.0001**
NG50	0.6507	**0.0005**
N75	0.7568	**0.0001**
# Contigs	0.7234	**0.0002**
# Misassemblies	**0.0352**	**0.0249**
# Mismatches per 100 kbp	0.8978	**0.0001**

4.2 Discussion

4.2.1 Assembler's Divergence

The first point we may discuss is about assemblers' divergence. Except for the last experiment, all others had differences between them. We could try to define that to assembling strategies, and it may be right, but even inside certain groups, as in the Bruijn assemblers, the strategies vary in some aspects. So, it is our first contribution: the assembler choice is important and will depend on the scientist's goal.

The high divergence of SSAKE results seen in all experiments was probably caused by the fact that it uses an Overlap-and-Extend Greedy Strategy, and was developed in the context of shorter reads (Warren et al. 2007). SSAKE requires error-free reads and had already been proved with a worse performance than traditional de Bruijn assemblers, in some features such as N50 and number of misassemblies (Paszkiewicz and Studholme 2010; Zerbino 2010; Simpson et al. 2009).

Generally, we eliminated some assemblers' outputs based on total length. In many cases, they were really larger than the reference genome. It was probably a consequence of incorrect repetitive regions in the assemblies, e.g. duplication genes. When analyzing QUAST results we found a higher duplication rate to those assemblies.

4.2.2 Sensitivity versus Specificity

However, we are not going far with assemblers subject. We will focus on parameters divergences. We have seen contiguity features that were influenced by all parameters. They demonstrated high sensitivity, in many cases with a linear regression coefficient, positively or negatively correlated to studied parameters. It has already been discussed that such features have high sensitivity, but lack specificity (Phillippy et al. 2008; Vezzi et al. 2012).

Imagine we know which assemblies have good quality and which do not (Figure 4.15). When we say contiguity features have high sensitivity, we mean they correctly identify good quality assemblies when they truly have good quality. Moreover, when we say they lack specificity, we mean they wrongly identify bad quality assemblies as good quality.

When N50, for example, grows as read length grows, and we consider bigger N50 values as a synonym of good quality, we can correctly detect a

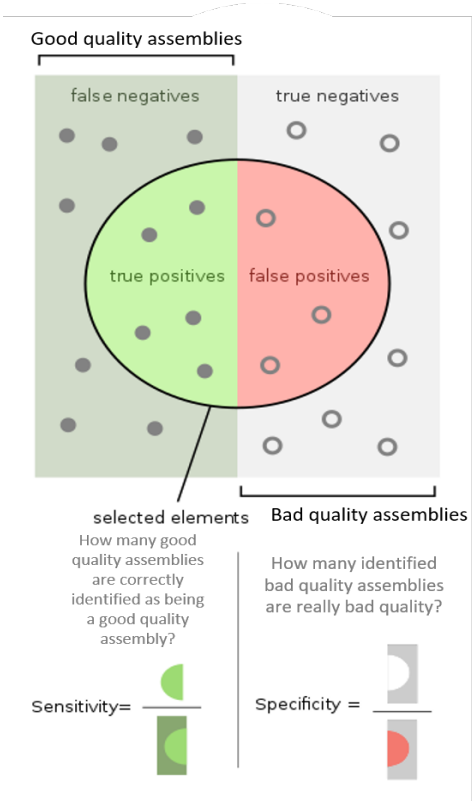

Figure 4.15: Sensitivity versus Specificity. Modified from Wikipedia

good quality assembly, hence larger reads probably generate better assemblies. But we also may mistake, since we could have a contiguous assembly with a lot of incorrect bases detection. That is why other high specificity features are necessary.

In addition, those contiguity features are redundant. They all vary together. That was Vezzi (Vezzi et al. 2012) issue at his work, firstly to generate new synthetic features through PCA (Principal Component Analysis), and secondly to detect most informative features through ICA (Independent Component Analysis); in order to eliminate redundancy, and avoid biases when all features are considered together. However, his work at that time just analyzed a few features and did not get high specificity as well.

Maybe the genomic features (Complete and Partial BUSCO, GC Content, Number of Predicted Genes) could confer that needed specificity. We have seen they changing a little, only in specific situations. Additional experiments and further studies are necessary to test this hypothesis. Also, more features should be included in our Feature Space Analysis.

Indeed, we may conclude that GAAF provided enough tools and methods, attending initial requisites, to pursue responding to our motivation research in Genome Assembly Quality subject. The next chapter briefly describes

the next steps.

5
Conclusions

The current work proposed to develop a Framework for the Domain of Feature Analysis of Genome Assemblies. It has four main control modules, capable of generating artificial reads, assembling, assembly evaluating and statistics analyzing.

5.1
Main Contributions

Firstly, we contributed to the field through the Systematic Mapping, listing the main proposed features, and classifying them into three groups: Contiguity, Base Analysis, and Genomic Analysis. Secondly, the development of GAAF contributed mainly to:

1. Reanalyzing entire experiments based on previous data;

2. Adding easily new tools (read generation software, features, assemblers) to the instantiated framework;

3. Abstraction;

4. Reproducibility.

The Framework contributes to the field in the way it is used and instantiated. In comparison to a simple script which calls tools, GAAF creates the possibility to reanalyze entire experiments, without running everything from point zero. It means that when an error occurs, or when the experiment is interrupted in the middle, one may continue running GAAF from where it was in the process; beyond the fact that one may add a new assembler and reanalyze the experiment considering the assemblies from that assembler, in conjunction with the assemblies and features generated previously.

However, the greatest advantage is the easier practice of adding new tools without previous knowledge about other GAAF components. It is just necessary to extend an abstract class, and to create tool-specific functions.

In addition, logging support, config file, and complete report give user enough experiment details to be easily reproduced by others.

Related to the case study, we may briefly list the following conclusions:

1. Sensitivity detection in contiguity features;

2. BUSCO features as indicators of low coverage and low phred quality;

3. The potential influence of repetitive regions over contiguity, base analysis and genomic features;

Initial studies in *Escherichia coli* have demonstrated how GAAF may be used, and the conclusions we may take through it. We have seen that contiguity features are highly sensitive, and varied according to read length, coverage, phred quality and the number of repetitive regions [1].

5.2 Future and Ongoing Work

Now, many next steps emerge in GAAF development and in Feature Analysis of Genome Assemblies.

For *Escherichia coli*, new parameters, e.g. k-mer size, and genomic characteristics, e.g. duplication and mutation rates, need to be analyzed, as well as more features. Then, other species and taxonomic groups may be analyzed in order to create a better comprehension of the subject. Also, other sequencing technologies and assembly levels, such as scaffolds, shall be contemplated.

In terms of Framework development, we may upgrade and test new modules, making them more robust and even more flexible to attend as many different sequencing and assembly technologies as possible.

These new modules could include a graphic interface, giving users experience more importance. This interface might include Experiment Design, Process Status, Process Manipulation, and Data Access. Also, it is very important to create a knowledge base, persisting experiments data, through a DBMS (Database Management System). It will reinforce the possibility to reanalyze and upgrade previous experiments.

[1] and other genomic characteristics

6
Bibliographic References

[Abante et al. 2017] Abante, J. et al. (2017). HiMMe: using genetic patterns as a proxy for genome assembly reliability assessment. *BMC Genomics*, 18(1):694.

[Angly et al. 2012] Angly, F. E., Willner, D., Rohwer, F., Hugenholtz, P., and Tyson, G. W. (2012). Grinder: a versatile amplicon and shotgun sequence simulator. *Nucleic acids research*, 40(12):e94.

[Bankevich et al. 2012] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., and Pevzner, P. A. (2012). SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology : a journal of computational molecular cell biology*, 19(5):455–77.

[Barthelson et al. 2011] Barthelson, R. et al. (2011). Plantagora : Modeling Whole Genome Sequencing and Assembly of Plant Genomes. 6(12).

[Batzer and Deininger 2002] Batzer, M. A. and Deininger, P. L. (2002). Alu repeats and human genomic diversity. *Nature Reviews Genetics*, 3(5):370–379.

[Bergeron et al. 2006] Bergeron, A., Mixtacki, J., and Stoye, J. (2006). A Unifying View of Genome Rearrangements. *Springer-Verlag Berlin Heidelberg 2006*.

[Bradnam et al. 2013] Bradnam et al. (2013). Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1):10.

[Castro and Ng 2017] Castro, C. J. and Ng, T. F. F. (2017). U50: A New Metric for Measuring Assembly Output Based on Non-Overlapping, Target-Specific Contigs. *Journal of computational biology : a journal of computational molecular cell biology*, 24(11):1071–1080.

[Chen et al. 2017] Chen, Q., Lan, C., Zhao, L., Wang, J., Chen, B., and Chen, Y.-P. P. (2017). Recent advances in sequence assembly: principles and applications. *Briefings in Functional Genomics*, 16(6):361–378.

[Chevreux et al. 2004] Chevreux, B., Pfisterer, T., Drescher, B., Driesel, A. J., Müller, W. E. G., Wetter, T., and Suhai, S. (2004). Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. *Genome research*, 14(6):1147–59.

[Chikhi and Medvedev 2013] Chikhi, R. and Medvedev, P. (2013). Informed and automated k -mer size selection for genome assembly. *Bioinformatics*, 30(1):31–37.

[Christopher M. Bishop 2006] Christopher M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.

[Clarke et al. 2009] Clarke, J., Wu, H.-C., Jayasinghe, L., Patel, A., Reid, S., and Bayley, H. (2009). Continuous base identification for single-molecule nanopore DNA sequencing. *Nature Nanotechnology*, 4(4):265–270.

[Commins et al. 2009] Commins, J., Toft, C., and Fares, M. A. (2009). Computational biology methods and their application to the comparative genomics of endocellular symbiotic bacteria of insects. *Biological procedures online*, 11:52–78.

[Consortium 2004] Consortium, I. H. G. S. (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431:931 – 945.

[Darling et al. 2011] Darling, A. E. et al. (2011). Mauve Assembly Metrics. *Bioinformatics*, 27(19):2756–2757.

[Delcher et al. 2007] Delcher, A. L., Bratke, K. A., Powers, E. C., and Salzberg, S. L. (2007). Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, 23(6):673–679.

[Denton et al. 2014] Denton, J. F. et al. (2014). Extensive Error in the Number of Genes Inferred from Draft Genome Assemblies. *PLoS Computational Biology*, 10(12):e1003998.

[Dohm et al. 2007] Dohm, J. C., Lottaz, C., Borodina, T., and Himmelbauer, H. (2007). SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17(11):1697–1706.

[Dunnett 1955] Dunnett, C. W. (1955). A Multiple Comparison Procedure for Comparing Several Treatments with a Control. *Journal of the American Statistical Association*, 50(272):1096–1121.

[Earl et al. 2011] Earl et al. (2011). Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome research*, 21(12):2224–41.

[Eisenstein 2012] Eisenstein, M. (2012). Oxford Nanopore announcement sets sequencing sector abuzz. *Nature Biotechnology*, 30(4):295–296.

[Escalona et al. 2016] Escalona, M., Rocha, S., and Posada, D. (2016). A comparison of tools for the simulation of genomic next-generation sequencing data. *Nature reviews. Genetics*, 17(8):459–69.

[Fayad and Schmidt 1997] Fayad, M. and Schmidt, D. (1997). Object-Oriented Application Frameworks Naval Open Systems Architecture Strategy View project Unified Software Engines (USEs): A unified approach to building software systems View project. *Article in Communications of the ACM*.

[Fedurco et al. 2006] Fedurco, M., Romieu, A., Williams, S., Lawrence, I., and Turcatti, G. (2006). BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies. *Nucleic acids research*, 34(3):e22.

[Greenleaf and Sidow 2014] Greenleaf, W. J. and Sidow, A. (2014). The future of sequencing: convergence of intelligent design and market Darwinism. *Genome biology*, 15(3):303.

[Gur-Arie et al. 2000] Gur-Arie, R., Cohen, C. J., Eitan, Y., Shelef, L., Hallerman, E. M., and Kashi, Y. (2000). Simple sequence repeats in Escherichia coli: abundance, distribution, composition, and polymorphism. *Genome research*, 10(1):62–71.

[Gurevich et al. 2013] Gurevich, A. et al. (2013). QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075.

[Haque et al. 2013] Haque, F., Li, J., Wu, H.-C., Liang, X.-J., and Guo, P. (2013). Solid-state and biological nanopore for real-time sensing of single chemical and sequencing of DNA. *Nano Today*, 8(1):56–74.

[Harris et al. 2008] Harris, T. D., Buzby, P. R., Babcock, H., Beer, E., Bowers, J., Braslavsky, I., Causey, M., Colonell, J., DiMeo, J., Efcavitch, J. W., Giladi, E., Gill, J., Healy, J., Jarosz, M., Lapen, D., Moulton, K., Quake, S. R., Steinmann, K., Thayer, E., Tyurina, A., Ward, R., Weiss, H., and Xie, Z. (2008). Single-Molecule DNA Sequencing of a Viral Genome. *Science*, 320(5872):106–109.

[Heather and Chain 2016] Heather, J. M. and Chain, B. (2016). The sequence of sequencers: The history of sequencing DNA. *Genomics*, 107(1):1–8.

[Hernandez et al. 2008] Hernandez, D., François, P., Farinelli, L., Osterås, M., and Schrenzel, J. (2008). De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome research*, 18(5):802–9.

[Hu et al. 2012] Hu, X., Yuan, J., Shi, Y., Lu, J., Liu, B., Li, Z., Chen, Y., Mu, D., Zhang, H., Li, N., Yue, Z., Bai, F., Li, H., and Fan, W. (2012). pIRS: Profile-based Illumina pair-end reads simulator. *Bioinformatics*, 28(11):1533–1535.

[Hunt et al. 2013] Hunt, M. et al. (2013). REAPR: a universal tool for genome assembly evaluation. *Genome Biology*, 14(5):R47.

[Idury and Waterman 1995] Idury, R. M. and Waterman, M. S. (1995). A New Algorithm for DNA Sequence Assembly. *Journal of Computational Biology*, 2(2):291–306.

[Illumina 2011] Illumina (2011). Quality Scores for Next-Generation Sequencing. Technical report.

[Illumina 2014] Illumina (2014). Estimating Sequencing Coverage. Technical report.

[Illumina 2019] Illumina (2019). Sequencing Platforms | Compare NGS platform applications & specifications.

[Jackman et al. 2017] Jackman, S. D., Vandervalk, B. P., Mohamadi, H., Chu, J., Yeo, S., Hammond, S. A., Jahesh, G., Khan, H., Coombe, L., Warren, R. L., and Birol, I. (2017). ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. *Genome research*, 27(5):768–777.

[Jackson et al. 1972] Jackson, D. A., Symons, R. H., and Berg, P. (1972). Biochemical method for inserting new genetic information into DNA of Simian Virus 40: circular SV40 DNA molecules containing lambda phage genes and the galactose operon of Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America*, 69(10):2904–9.

[Jeck et al. 2007] Jeck, W. R., Reinhardt, J. A., Baltrus, D. A., Hickenbotham, M. T., Magrini, V., Mardis, E. R., Dangl, J. L., and Jones, C. D. (2007). Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23(21):2942–2944.

[Karlsson et al. 2015] Karlsson, E., Lärkeryd, A., Sjödin, A., Forsman, M., and Stenberg, P. (2015). Scaffolding of a bacterial genome using MinION nanopore sequencing. *Scientific Reports*, 5(1):11996.

[Kaufmann and Schering 2014] Kaufmann, J. and Schering, A. (2014). Analysis of Variance ANOVA. In *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, Chichester, UK.

[Koonin and Galperin 2003] Koonin, E. and Galperin, M. (2003). Genome Annotation and Analysis. In *Sequence - Evolution - Function: Computational Approaches in Comparative Genomics*, chapter 5. Kluwer Academic, Boston, USA.

[Kruskal and Wallis 1952] Kruskal, W. H. and Wallis, W. A. (1952). Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583.

[Land et al. 2014] Land, M. L. et al. (2014). Quality scores for 32,000 genomes. *Standards in Genomic Sciences*, 9(1):20.

[Landin and Niklasson 1995] Landin, N. and Niklasson, A. (1995). *Development of Object-Oriented Frameworks*.

[Lewin 2009] Lewin, B. (2009). *Genes IX*. Artmed, 9 edition.

[Liu et al. 2018] Liu, D., Hunt, M., and Tsai, I. J. (2018). Inferring synteny between genome assemblies: A systematic evaluation. *BMC Bioinformatics*, 19(1):1–6.

[Lo et al. 2013] Lo, C. et al. (2013). Evaluating genome architecture of a complex region via generalized bipartite matching. *BMC Bioinformatics*, 14(Suppl 5):S13.

[Loman et al. 2012] Loman, N. J., Misra, R. V., Dallman, T. J., Constantinidou, C., Gharbia, S. E., Wain, J., and Pallen, M. J. (2012). Performance comparison of benchtop high-throughput sequencing platforms. *Nature Biotechnology*, 30(5):434–439.

[Loman and Quinlan 2014] Loman, N. J. and Quinlan, A. R. (2014). Poretools: a toolkit for analyzing nanopore sequence data. *Bioinformatics (Oxford, England)*, 30(23):3399–401.

[Lukashin and Borodovsky 1998] Lukashin, A. V. and Borodovsky, M. (1998). GeneMark.hmm: new solutions for gene finding. *Nucleic acids research*, 26(4):1107–15.

[Madoui et al. 2015] Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015). Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC Genomics*, 16(1):327.

[Magoc et al. 2013] Magoc, T., Pabinger, S., Canzar, S., Liu, X., Su, Q., Puiu, D., Tallon, L. J., and Salzberg, S. L. (2013). GAGE-B: an evaluation of genome assemblers for bacterial organisms. *Bioinformatics (Oxford, England)*, 29(14):1718–25.

[Majoros et al. 2004] Majoros, W. H., Pertea, M., and Salzberg, S. L. (2004). TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. *Bioinformatics*, 20(16):2878–2879.

[Markiewicz and Lucena 2000] Markiewicz, M. E. and Lucena, C. J. P. (2000). Understanding Object-Oriented Framework Engineering. Technical report, PUC-Rio.

[McKernan et al. 2009] McKernan, K. J., Peckham, H. E., Costa, G. L., McLaughlin, S. F., Fu, Y., Tsung, E. F., Clouser, C. R., Duncan, C., Ichikawa, J. K., Lee, C. C., Zhang, Z., Ranade, S. S., Dimalanta, E. T., Hyland, F. C., Sokolsky, T. D., Zhang, L., Sheridan, A., Fu, H., Hendrickson, C. L., Li, B., Kotler, L., Stuart, J. R., Malek, J. A., Manning, J. M., Antipova, A. A., Perez, D. S., Moore, M. P., Hayashibara, K. C., Lyons, M. R., Beaudoin, R. E., Coleman, B. E., Laptewicz, M. W., Sannicandro, A. E., Rhodes, M. D., Gottimukkala, R. K., Yang, S., Bafna, V., Bashir, A., MacBride, A., Alkan, C., Kidd, J. M., Eichler, E. E., Reese, M. G., De La Vega, F. M., and Blanchard, A. P. (2009). Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. *Genome research*, 19(9):1527–41.

[Mikheenko et al. 2016] Mikheenko, A. et al. (2016). Icarus: visualizer for de novo assembly evaluation. *Bioinformatics*, 32(21):3321–3323.

[Miller et al. 2010] Miller, J. R., Koren, S., and Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–27.

[Moriel et al. 2012] Moriel, D. G., Rosini, R., Seib, K. L., Serino, L., Pizza, M., and Rappuoli, R. (2012). Escherichia coli: great diversity around a common core. *mBio*, 3(3).

[Myers 2005] Myers, E. W. (2005). The fragment assembly string graph. *Bioinformatics*, 21(Suppl 2):ii79–ii85.

[Nemenyi 1963] Nemenyi, P. (1963). *Distribution-free multiple comparisons.* PhD thesis, Princeton University.

[Nowoshilow et al. 2018] Nowoshilow, S., Schloissnig, S., Fei, J.-F., Dahl, A., Pang, A. W. C., Pippel, M., Winkler, S., Hastie, A. R., Young, G., Roscito, J. G., Falcon, F., Knapp, D., Powell, S., Cruz, A., Cao, H., Habermann, B., Hiller, M., Tanaka, E. M., and Myers, E. W. (2018). The axolotl genome and the evolution of key tissue formation regulators. *Nature*, 554(7690):50–55.

[Nyrén and Lundin 1985] Nyrén, P. and Lundin, A. (1985). Enzymatic method for continuous monitoring of inorganic pyrophosphate synthesis. *Analytical Biochemistry*, 151(2):504–509.

[Paszkiewicz and Studholme 2010] Paszkiewicz, K. and Studholme, D. J. (2010). De novo assembly of short sequence reads. *Briefings in Bioinformatics*, 11(5):457–472.

[Pevzner et al. 2001] Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753.

[Phillippy et al. 2008] Phillippy, A. M., Schatz, M. C., and Pop, M. (2008). Genome assembly forensics: finding the elusive mis-assembly. *Genome Biology*, 9(3).

[Pop 2009] Pop, M. (2009). Genome assembly reborn: recent computational challenges. *Briefings in Bioinformatics*, 10(4):354.

[Quail et al. 2012] Quail, M., Smith, M. E., Coupland, P., Otto, T. D., Harris, S. R., Connor, T. R., Bertoni, A., Swerdlow, H. P., and Gu, Y. (2012). A tale of three next generation sequencing platforms: comparison of Ion torrent, pacific biosciences and illumina MiSeq sequencers. *BMC Genomics*, 13(1):341.

[Rahman and Pachter 2013] Rahman, A. and Pachter, L. (2013). CGAL: computing genome assembly likelihoods. *Genome Biology*, 14(1):R8.

[Ronaghi et al. 1998] Ronaghi, M., Uhlén, M., and Nyrén, P. (1998). A Sequencing Method Based on Real-Time Pyrophosphate. *Science*, 281(5375):363–365.

[Rothberg et al. 2011] Rothberg, J. M., Hinz, W., Rearick, T. M., Schultz, J., Mileski, W., Davey, M., Leamon, J. H., Johnson, K., Milgrew, M. J., Edwards, M., Hoon, J., Simons, J. F., Marran, D., Myers, J. W., Davidson, J. F., Branting, A., Nobile, J. R., Puc, B. P., Light, D., Clark, T. A., Huber, M., Branciforte, J. T., Stoner, I. B., Cawley, S. E., Lyons, M., Fu, Y., Homer, N., Sedova, M., Miao, X., Reed, B., Sabina, J., Feierstein, E., Schorn, M., Alanjary, M., Dimalanta, E., Dressman, D., Kasinskas, R., Sokolsky, T., Fidanza, J. A., Namsaraev, E., McKernan, K. J., Williams, A., Roth, G. T., and Bustillo, J. (2011). An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356):348–352.

[Saiki et al. 1988] Saiki, R., Gelfand, D., Stoffel, S., Scharf, S., Higuchi, R., Horn, G., Mullis, K., and Erlich, H. (1988). Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase. *Science*, 239(4839):487–491.

[Salzberg et al. 2012] Salzberg, S. L. et al. (2012). GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, 22(3):557–567.

[Sanger et al. 1977a] Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, J. C., Hutchison, C. A., Slocombe, P. M., and Smith, M. (1977a). Nucleotide sequence of bacteriophage ϕX174 DNA. *Nature*, 265(5596):687–695.

[Sanger and Coulson 1975] Sanger, F. and Coulson, A. (1975). A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*, 94(3):441–448.

[Sanger et al. 1977b] Sanger, F., Nicklen, S., and Coulson, A. R. (1977b). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 74(12):5463–7.

[Schadt et al. 2010] Schadt, E. E., Turner, S., and Kasarskis, A. (2010). A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2):R227–R240.

[Seitz et al. 2018] Seitz, A., Hanssen, F., and Nieselt, K. (2018). DACCOR-Detection, characterization, and reconstruction of repetitive regions in bacterial genomes. *PeerJ*.

[Shapiro and Wilk 1965] Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611.

[Shendure and Ji 2008] Shendure, J. and Ji, H. (2008). Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145.

[Simão et al. 2015] Simão, F. A. et al. (2015). BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, 31(19):3210–3212.

[Simpson et al. 2009] Simpson, J. T., Wong, K., Jackman, S. D., Simpson, J. T., Wong, K., Jackman, S. D., Schein, J. E., and Jones, S. J. M. (2009). ABySS : A parallel assembler for short read sequence data ABySS : A parallel assembler for short read sequence data. pages 1117–1123.

[Sims et al. 2014] Sims, D., Sudbery, I., Ilott, N. E., Heger, A., and Ponting, C. P. (2014). Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics*, 15(2):121–132.

[Snustad and Simmons 2012] Snustad, D. P. and Simmons, M. J. (2012). *Fundamentos de Genética*. Guanabara Koogan, 4 edition.

[Staden 1980] Staden, R. (1980). A new computer method for the storage and manipulation of DNA gel reading data. *Nucleic acids research*, 8(16):3673–94.

[Treangen and Salzberg 2012] Treangen, T. J. and Salzberg, S. L. (2012). Repetitive DNA and next-generation sequencing : computational challenges and solutions. 13(JANUARY).

[van Dijk et al. 2014] van Dijk, E. L., Auger, H., Jaszczyszyn, Y., and Thermes, C. (2014). Ten years of next-generation sequencing technology. *Trends in Genetics*, 30(9):418–426.

[Vezzi et al. 2012] Vezzi, F., Narzisi, G., and Mishra, B. (2012). Feature-by-Feature – Evaluating De Novo Sequence Assembly. *PLoS ONE*, 7(2):e31002.

[Warren et al. 2007] Warren, R. L., Sutton, G. G., Jones, S. J. M., and Holt, R. A. (2007). Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501.

[Waterhouse et al. 2017] Waterhouse, R. M. et al. (2017). BUSCO Applications from Quality Assessments to Gene Prediction and Phylogenomics. *Mol. Biol. Evol. 35(3):543–548*, 35(3):543–548.

[Yandell and Ence 2012] Yandell, M. and Ence, D. (2012). A beginner's guide to eukaryotic genome annotation. *Nature Publishing Group*, 13.

[Zerbino 2010] Zerbino, D. R. (2010). Using the Velvet de novo assembler for short-read sequencing technologies. *Current protocols in bioinformatics*, Chapter 11:Unit 11.5.

[Zerbino and Birney 2008] Zerbino, D. R. and Birney, E. (2008). Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829.

[Zimin et al. 2013] Zimin, A. V., Marçais, G., Puiu, D., Roberts, M., Salzberg, S. L., and Yorke, J. A. (2013). The MaSuRCA genome assembler. *Bioinformatics*, 29(21):2669–2677.

A
Glossary

Genome: according to the Genetics Home Reference (ghr.nlm.nih.gov), Genome is "an organism's complete set of DNA, including all of its genes. Each genome contains all of the information needed to build and maintain that organism. In humans, a copy of the entire genome—more than 3 billion DNA base pairs—is contained in all cells that have a nucleus".

Gene: according to the Genetics Home Reference (ghr.nlm.nih.gov), a "gene is the basic physical and functional unit of heredity. Genes are made up of DNA. Some genes act as instructions to make molecules called proteins. However, many genes do not code for proteins".

Read: is one sequence read/generated by the sequencing machine.

Contig: derives from the word contiguous, and means a set of overlapping DNA fragments (reads) that together produce a consensus region of DNA(Staden 1980).

Scaffold: is a series of contigs separated by gaps of known length.

According to QUAST Manual, the selected features are described as follows:

Complete/Partial BUSCO (%): is the percent of BUSCO genes found in the assembly completely (or partially).

 # predicted genes: is the number of genes in the assembly found by GlimmerHMM (Delcher et al. 2007).

Largest alignment: is the length of the largest continuous alignment in the assembly. A value can be smaller than a value of largest contig if the largest contig is misassembled or partially unaligned.

 # contigs: is the total number of contigs in the assembly.

Largest contig: is the length of the longest contig in the assembly.

Total length: is the total number of bases in the assembly.

Total aligned length: is the total number of aligned bases in the assembly. A value is usually smaller than a value of total length because some of the contigs may be unaligned or partially unaligned.

N50: is the length for which the collection of all contigs of that length or longer covers at least half an assembly.

NG50: is the length for which the collection of all contigs of that length or longer covers at least half the reference genome. This metric is computed only if the reference genome is provided.

N75 and NG75: are defined similarly to N50 but with 75

L50 (L75, LG50, LG75): is the number of contigs equal to or longer than N50 (N75, NG50, NG75) In other words, L50, for example, is the minimal number of contigs that cover half the assembly.

NA50, NGA50, NA75, NGA75, LA50, LA75, LGA50, LGA75 : ("A" stands for "aligned") are similar to the corresponding metrics without "A", but in this case aligned blocks instead of contigs are considered. Aligned blocks are obtained by breaking contigs at misassembly events and removing all unaligned bases.

GC (%): is the total number of G and C nucleotides in the assembly, divided by the total length of the assembly.

 # misassemblies:is the number of positions in the contigs (breakpoints) that satisfy one of the following criteria: the left flanking sequence aligns over 1 kbp away from the right flanking sequence on the reference; flanking sequences overlap on more than 1 kbp; flanking sequences align to different strands or different chromosomes;

 # mismatches per 100 kbp: is the average number of mismatches per 100000 aligned bases. True SNPs and sequencing errors are not distinguished and are counted equally.

Avg. coverage depth: is the average depth of coverage

Mapped (%): is the percentage of reads that mapped to the assembly

Reference Mapped (%): is the percentage of reads that mapped to the reference genome