MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA
CAMPUS SEABRA

GUSTAVO JORGE NOVAES SILVA

FERRAMENTAS COMPUTACIONAIS PARA A MONTAGEM DE GENOMAS BACTERIANOS: UM ESTUDO COMPARATIVO

Seabra – BA

10 de março de 2021

GUSTAVO JORGE NOVAES SILVA

FERRAMENTAS COMPUTACIONAIS PARA A MONTAGEM DE GENOMAS BACTERIANOS: UM ESTUDO COMPARATIVO

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Informática do Instituto Federal de Educação, Ciência e Tecnologia da Bahia – Campus Seabra, como requisito parcial para obtenção do diploma de Técnico em Informática.

Orientador: Prof. MSc Matheus Brito de Oliveira

Seabra – BA

10 de março de 2021

Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)

Coordenação do Curso Técnico em Informática

Campus Seabra

GUSTAVO JORGE NOVAES SILVA

Este Trabalho de conclusão de Curso foi julgado adequado para a obtenção do título de Técnico em Informática , sendo aprovado pela Coordenação do curso Técnico em Informática do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Campus Seabra.

Banca examinadora:

Orientador: Prof. MSc Matheus Brito de Oliveira
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)

Prof. MSc. João Paulo Pereira de Almeida
Universidade Federal de Minas Gerais (UFMG)

Profa. MSc. Mayane Santos Amorim
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)

Seabra – BA

10 de março de 2021

Resumo

O surgimento das tecnologias de sequenciamento de próxima geração (NGS) transformou radicalmente as técnicas de identificação das bases nitrogenadas do DNA e potencializou a produção de pesquisas científicas por todo o globo, sobretudo no campo do sequenciamento e análise gênica de bactérias. A etapa posterior ao sequenciamento - a da montagem computacional dos fragmentos de DNA - é um processo complexo e altamente dependente das plataformas em que os organismos foram sequenciados. Essa dependência, vinculada a diversos outros fatores, acabou criando condições favoráveis para uma produção em larga escala de montadores de genomas bacterianos, em que cada um possui configurações e parâmetros distintos de funcionamento. Para mais, a escolha dessa ferramenta é de suma importância para o sucesso da pesquisa em desenvolvimento. Entretanto, a escassez de trabalhos atuais que discutam sobre o desempenho desses *softwares* torna a escolha de um montador algo difícil de ser realizado. Diante o exposto, este trabalho propõe-se a desenvolver um estudo comparativo com 7 ferramentas comumente utilizadas para a montagem de genomas bacterianos, fornecendo informações relevantes sobre o desempenho, implementação e usabilidade de cada uma delas. Todos os montadores foram testados com as mesmas amostras biológicas, obtidas no repositório *Sequence Read Archive* (SRA) e oriundas da plataforma de sequenciamento de segunda geração *Ion Personal Genome Machine* (IonPGM). Em nosso estudo buscamos descrever detalhadamente de que forma os recursos e estratégias de montagem adotadas por cada programa influenciaram em seu desempenho e na qualidade final dos seus resultados. Para isso, utilizamos diferentes métricas e parâmetros de qualidade, resultando em mais de 200 montagens. Em suma, pretendemos com esse trabalho fornecer aos profissionais de bioinformática informações suficientes para guiá-los na escolha da ferramenta adequada aos seus projetos, contribuindo, dessa maneira, para o avanço das técnicas relacionadas à montagem de genomas bacterianos e na elaboração de pesquisas científicas na área.

Palavras-chave: montagem de genomas, bacterias, estudo comparativo, NGS, bioinformática.

Abstract

The emergence of next generation sequencing technologies (NGS) has radically transformed the techniques for identifying the nitrogenous bases of DNA and has boosted the production of scientific research across the globe, especially in the field of sequencing and genetic analysis of bacteria. The post-sequencing step - the computational assembly of DNA fragments - is a complex process and highly dependent on the platforms on which the organisms were sequenced. This dependence, linked to several other factors, ended up creating favorable conditions for a large-scale production of assemblers of bacterial genomes, in which each one has different configurations and operating parameters. Furthermore, the choice of this tool is of paramount importance for the success of research in development. However, the scarcity of current works that deal with the performance of these software makes the choice of an assembler something difficult to be done. Given the above, this work proposes to develop a comparative study with 7 tools commonly used for the assembly of bacterial genomes, providing relevant information about the performance, implementation and usability of each one. All assemblers were tested with the same biological samples, obtained from the Sequence Read Archive (SRA) repository and from the second generation sequencing platform Ion Personal Genome Machine (IonPGM). In our study we tried to describe in detail how the assembly resources and strategies adopted by each program influenced its performance and the final quality of its results. For that, we use different metrics and quality parameters, resulting in more than 200 assemblies. In short, with this work we intend to provide bioinformatics professionals with sufficient information to guide them in choosing the appropriate tool for their projects, thus contributing to the advancement of techniques related to the assembly of bacterial genomes and in the development of scientific research in the area.

Keywords: assembly of genomes, bacteria, comparative study, NGS, bioinformatics.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DBG	de Bruijn Graph
DNA	Deoxyribonucleic Acid
IonPGM	Ion Personal Genome Machine
NCBI	National Center for Biotechnology Information
NGS	Next Generation Sequencing
OLC	Overlap Layout Consensus
RAM	Random Access Memory
SRA	Sequence Read Archive
SMS	Single Molecule Sequencing
WGS	Whole Genome Shotgun

Sumário

1 INTRODUÇÃO

Durante o século XX os cientistas James Watson, Francis Crick e Maurice Wilkins foram responsáveis pelo o que atualmente é considerado uma das conquistas científicas mais relevantes da história: a descoberta da estrutura de dupla hélice do DNA (ácido desoxirribonucleico) em 1953 (RAMB, 2005). Essa pesquisa rendeu aos cientistas o prêmio Nobel de Fisiologia ou Medicina em 1962 e abriu o caminho para o desenvolvimento de diversas pesquisas dentro do ramo da genética, especialmente em áreas como a transgenia, tratamentos de doenças hereditárias, clonagem, manipulação controlada do DNA, predição genética e diversas outras, transparecendo à comunidade científica a relevância de se ter acesso à sequência genética de um determinado organismo (SCHEID; DELIZOICOV; FERRARI, 2003; MOORE, 2004; HEPP; NONOHAY, 2016).

Dado a importância e a grande expectativa dos estudos na área, diversos campos se desenvolveram concomitantemente à biologia molecular com o fito de criar métodos e técnicas que aumentassem a eficácia de suas pesquisas. Assim sendo, a última década do século XX foi marcada por grandes avanços nos estudos moleculares que, acompanhando o surgimento de novos sistemas computacionais, acabou criando condições favoráveis para a eclosão de uma nova ciência: a bioinformática (SOLÓRZANO et al., 2003). Segundo Alves (2013), o mais novo campo de pesquisa pode ser definido como uma área interdisciplinar entre a biologia e a ciência da computação que abrange conceitos de diversas áreas do conhecimento, como a matemática, física, estatística e a medicina. De modo geral, a bioinformática destina-se ao desenvolvimento e a aplicação de métodos e ferramentas computacionais para solucionar problemas da biologia molecular, envolvendo os processos de organização, armazenamento, gestão, análise e interpretação de informações biológicas complexas (LIMA, 2007; ALVES, 2013).

O escopo de atuação da bioinformática é bastante extenso, englobando áreas como o sequenciamento e a montagem de genomas, anotação genômica, predição de genes, mapeamento de genomas e diversas outras (OLIVEIRA, 2017). De acordo com Lima (2007), o sequenciamento e a montagem de genomas podem ser classificados como a primeira etapa de um processo maior: a conclusão do sequenciamento completo de um genoma, já que após esse estágio há também os processos de anotação genômica e análise final.

O sequenciamento de DNA pode ser definido como um conjunto de técnicas laboratoriais que visam determinar a sequência exata das bases nitrogenadas presentes em uma molécula de ácido desoxirribonucleico, fornecendo, assim, informações valiosas sobre o funcionamento do genoma em questão (BROWN, 2002). Apesar dos inúmeros avanços

alcançados nos últimos anos, os métodos atuais para sequenciar o DNA não são capazes de processar as suas bases em ordem e sequenciá-las em uma única etapa. A justificativa para isso é o tamanho dos genomas, que costumam ser grandes e acabam impossibilitando o seu processamento de uma única vez. Dessa maneira, para sequenciar um genoma, o DNA do organismo é fragmentado em pedaços menores e aleatórios que são então submetidos a processos que possam deduzir a ordem exata dos seus nucleotídeos. Assim, no final, tem-se como produto um conjunto de fragmentos sequenciados que precisam ser reordenados e colocados na ordem original. Para isso, os pedaços são enviados a *softwares* que, baseados em algoritmos computacionais, obtêm a sequência genômica original do organismo; essa etapa é chamada de montagem *de novo* do genoma (WETTERSTRAND, 2019).

Os processos que envolvem a biologia molecular podem ser desenvolvidos por meio de diferentes abordagens, a exemplo dos métodos *in vitro* e *in silico*. O primeiro refere-se aos experimentos feitos em ambientes controlados e supervisionados - como os laboratórios - enquanto o segundo trata dos processos feitos via simulações computacionais. Essas simulações tem por objetivo construir modelos para representar fenômenos naturais ou laboratoriais, para que assim, por meio de parâmetros diversos, seja possível analisar o comportamento do fenômeno retratado quando submetido a diferentes circunstâncias (VALLE, 2019).

A adoção de métodos *in silico* vem promovendo inúmeros benefícios para a biologia, sobretudo na redução do tempo para a execução de experimentos, na automatização de processos e na simplificação de experimentos complexos (OLIVEIRA, 2017). Dessa forma, as simulações computacionais possuem vantagens quando comparadas aos experimentos tradicionais, uma vez que estes são complexos e dispendiosos em termos de materiais, infraestrutura e pessoal qualificado (SAUNDERS, 2004; MODA, 2007). Entretanto, essas abordagens também podem ser utilizadas em conjunto. O processo de fragmentação aleatória para a realização do sequenciamento, por exemplo, é classificado como uma etapa *in vitro*, enquanto a montagem dos fragmentos é obtida por meio de métodos *in silico* (HUSEMANN, 2011).

Nos últimos anos a habilidade de sequenciar o DNA evoluiu muito e promoveu inúmeros avanços para a comunidade científica, permitindo o desenvolvimento de pesquisas e estudos em áreas como a medicina, agricultura, pecuária e na análise gênica de diversos organismos, especialmente das bactérias. O sequenciamento e a montagem de genomas bacterianos é uma área em ascensão e de grande relevância para a biologia molecular, uma vez que o estudo dos organismos procariotos vem se tornando essencial para áreas como a microbiologia clínica - com a testagem de antibióticos e outros medicamentos - e para a compreensão de uma série de fenômenos, principalmente aqueles associados a patologias e investigações epidemiológicas (DIDELOT et al., 2012; HEPP; NONOHAY, 2016).

Entretanto, Oliveira (2017) destaca alguns pontos problemáticos para aqueles que

almejam ingressar no campo da bioinformática, como o tempo demasiado para a execução de algumas etapas, conhecimento prévio na área computacional, tempo elevado para treinar profissionais entre outros. Ademais, podemos ainda citar um outro problema: o caos gerado pelo grande número de programas destinados a realizar a etapa da montagem do genoma.

A produção crescente dessas ferramentas ocorreu por diversos motivos, incluindo a própria dificuldade para a execução do processo, as inúmeras variáveis existentes e as diversas formas de se abordar o mesmo problema. Esse fenômeno gerou como consequência um cenário ainda mais complexo para o desenvolvimento de pesquisas na área, sobretudo pela existência de uma grande variedade de montadores com configurações, instruções e parâmetros distintos de funcionamento. Além disso, alguns desses *softwares* são de difícil instalação, exigem tempo de compreensão por causa dos manuais longos e, por vezes, o usuário não consegue atingir o resultado esperado, seja pelo conjunto de dados utilizados como entrada ou por desconhecer o fluxo de funcionamento do programa. Assim, além de se preocuparem com os complexos fatores biológicos, os bioinformatas precisam também possuir conhecimento técnico em informática para escolher a ferramenta ideal para o seu projeto. Entretanto, a escassez de trabalhos atuais que discutam sobre o desempenho, implementação e usabilidade desses *softwares* torna a escolha de um montador algo difícil de ser realizado.

Portanto, o desenvolvimento de um estudo que disponibilize tais informações de maneira prática e fácil constitui uma importante adoção para pesquisadores no campo da biologia computacional pois, além auxiliá-los na escolha da ferramenta mais apropriada para o seu projeto – poupando tempo e esforço –, também forneceria informações relevantes para o desenvolvimento de novas ferramentas computacionais.

1.1 Motivação

A bioinformática tornou-se uma disciplina independente na década de 1980, período em que começaram a ser desenvolvidos os primeiros algoritmos computacionais para manipular grandes volumes de dados biológicos (ALVES, 2013).

Embora seja uma ciência recente, a bioinformática vem sendo amplamente utilizada na execução das mais variadas tarefas, como na análise de sequências nucleotídicas e proteicas, anotação de genes e genomas, diagnóstico e rastreio de doenças genéticas, sequenciamento e montagem de genomas, genômica comparativa e diversas outras (OTTO et al., 2007; ALVES, 2013). Justamente por sua grande aplicabilidade, essa ciência vem sendo cada vez mais requisitada pela comunidade científica e por pesquisadores do mundo todo.

A partir do século XXI, a bioinformática passou a receber maiores impactos da área tecnológica, principalmente devido ao avanço, miniaturização e barateamento dos

computadores. A utilização de máquinas mais poderosas e menos custosas permitiu a resolução de problemas complexos que seriam impossíveis sem os recursos atualmente disponíveis (VERLI, 2014). Em vista disso, discorrer sobre novos modelos computacionais para superar as limitações existentes nessa área torna-se primordial para o seu avanço.

Para mais, essa atualização constante dos recursos computacionais promoveu um desenvolvimento acelerado dessa ciência, ao ponto de programas anteriormente utilizados em larga escala se tornarem rapidamente ineficientes ou até mesmo obsoletos quando comparados as novas ferramentas que estão sendo produzidas (VERLI, 2014). Assim sendo, essa área requer do seu praticante "uma constante atenção a novas abordagens, métodos, requerimentos e tendências" (VERLI, 2014, p. 8).

Diante o exposto, o desenvolvimento de trabalhos acadêmicos que busquem discutir práticas atuais de pesquisa se torna de extrema relevância para os profissionais da bioinformática, especialmente aqueles envolvidos com campo da montagem genômica.

Desse modo, além da própria relevância que essa ciência exerce sobre a comunidade científica, podemos destacar outros fatores motivacionais para a confecção deste estudo, como a ausência de informações práticas e atuais sobre os programas de montagem, a complexidade para desenvolver trabalhos na área e o tempo elevado aplicado por cientistas em pesquisas para escolher uma ferramenta que atenda às suas necessidades.

1.2 Descrição do problema

A montagem de genomas bacterianos é um processo complexo e altamente dependente da plataforma de sequenciamento em que os organismos foram obtidos, sendo, então, fortemente influenciado pelas limitações oriundas dessas tecnologias e pelas características dos genomas sequenciados (MARIANO, 2015; RAMOS et al., 2013; KIRCHER; KELSO, 2010; CERDEIRA et al., 2011).

Devido ao desenvolvimento acelerado de novas técnicas para o sequenciamento de DNA, existem atualmente diversas tecnologias disponíveis para esse fim, em que cada uma utiliza de metodologias e estratégias próprias para a obtenção dos dados biológicos. Assim sendo, as plataformas produzem padrões diferentes de sequências, variando em fatores como o tamanho dos fragmentos (curtos ou longos), padrão de saída e a quantidade de fragmentos gerados por execução (OLIVEIRA, 2017).

Ademais, os programas de montagem são altamente sensíveis a esses fatores, sendo desenvolvidos com módulos e parâmetros específicos para atender as particularidades e os padrões de dados biológicos produzidos por cada tecnologia. Porém, como são muitas as plataformas de sequenciamento existentes, as variáveis a serem consideradas durante a construção de um montador são numerosas, o que dificulta a sua concepção. Além

dos coeficientes relacionados a etapa de sequenciamento, os programas de montagem são projetados com abordagens distintas, existindo, por exemplo, diferentes algoritmos computacionais para a execução dessa tarefa.

Esse cenário fez com que diversas ferramentas fossem desenvolvidas a fim de lidar com os fatores complexos que envolvem o processo de montagem e produzir melhores resultados. Entretanto, embora muitas estratégias tenham sido propostas, não existe um consenso sobre a melhor abordagem a ser utilizada (MARIANO, 2015). Como consequência, esse fenômeno fez com que muitos profissionais ficassem inseguros durante a escolha de uma ferramenta de montagem para a execução da sua pesquisa.

Portanto, sob o contexto analisado, as perguntas que compõem o problema de pesquisa são: mediante a uma grande quantidade de *softwares* de montagem de genomas bacterianos que prometem resultados com alta qualidade e grande eficácia, quais deles realmente apresentam um bom desempenho? Além do mais, quais fatores devem ser levados em consideração na escolha de um montador?

1.3 Objetivos Geral e Específicos

Evidenciar, por meio de um estudo comparativo, o desempenho, implementação e usabilidade dos principais montadores de genomas bacterianos gratuitos e de código aberto *(open source)* atualmente disponíveis.

Em um sentido mais particular do trabalho, os objetivos específicos são:

- Discorrer sobre as etapas que envolvem o processo da montagem de genomas bacterianos;
- Compreender de que maneira os recursos, estratégias e algoritmos de montagem adotados por cada programa influenciaram em seu desempenho;
- Disponibilizar aos pesquisadores e bioinformatas informações sobre os principais montadores de genomas bacterianos;

1.4 Estrutura do trabalho

O seguinte trabalho é constituído por 5 capítulos e suas respectivas seções, sendo o primeiro deles destinado a introdução, motivação, problema de pesquisa e descrição dos objetivos.

O segundo capítulo aborda o referencial teórico feito para a confecção do estudo, sendo este dividido em duas seções. A primeira seção apresenta a prática do sequenciamento de DNA, abordando a sua definição, contexto histórico, relevância e aplicações. A segunda parte é destinada à montagem de genomas, incluindo o seu conceito, descrição detalhada

do seu processo, os fatores complexos relacionados, algoritmos computacionais atualmente disponíveis para a sua execução e a montagem por referência e *de novo*.

O terceiro capítulo descreve a metodologia, incluindo a descrição dos materiais utilizados e as etapas seguidas. Esse capítulo é subdividido em quatro seções, sendo a primeira destinada aos dados biológicos utilizados e a plataforma de sequenciamento em que estes foram produzidos. A segunda seção aborda sobre o processo de seleção dos programas que participaram do estudo, evidenciando os motivos pelos quais foram escolhidos e as suas respectivas características. A terceira seção apresenta as métricas de qualidade utilizadas para mensurar o desempenho dos montadores. Por fim, a última seção trata das especificações da máquina em que os testes foram realizados e o sistema operacional utilizado para as execuções.

O quarto capítulo expõe os resultados obtidos com a execução dos programas, sendo também dividido em seções. As quatro primeiras seções descrevem o comportamento dos *softwares* levando em consideração as métricas de qualidade, as quais são: número de *contigs*, N50, *total lenght* e tempo de execução. A quinta seção aborda o desempenho geral dos montadores, principalmente sobre o nível de influência que os fatores externos (dados biológicos) e internos (recursos, módulos e algoritmos de montagem) exerceram sobre eles. A última seção refere-se ao nível geral de usabilidade e implementação dos programas.

Por fim, mediante a investigação feita, o último capítulo aborda as conclusões obtidas com o estudo comparativo, levando em consideração os objetivos e os problemas de pesquisa previamente descritos.

2 REVISÃO BIBLIOGRÁFICA

2.1 Sequenciamento de DNA

Um genoma pode ser definido como o conjunto de todo o DNA contido no núcleo de uma célula. O DNA, por sua vez, nada mais é do que um composto orgânico formado pelo encadeamento de nucleotídeos, que são unidades constituídas por uma pentose, um fosfato e uma base nitrogenada. As bases nitrogenadas podem ser de quatro tipos: Adenina, Citosina, Guanina e Timina (abreviados respectivamente como A, C, G, T). Ademais, as informações biológicas – como aquelas responsáveis pela síntese de proteínas - estão contidas na molécula de DNA e codificadas na ordem de suas bases (WETTERSTRAND, 2019; GÓES; OLIVEIRA, 2014; ANDRADE; CALDEIRA, 2009). Assim sendo, compreender o modo como tais bases estão dispostas na estrutura celular do organismo é de fundamental importância para acessar as suas informações.

Mediante o exposto, o sequenciamento de DNA pode ser definido como um conjunto de técnicas laboratoriais que visam determinar a sequência exata das bases nitrogenadas presentes no ácido desoxirribonucleico (BROWN, 2002). Devido a sua relevância, a habilidade de sequenciar o DNA transformou-se em um método aclamado pela comunidade científica e de suma importância para a biologia molecular, principalmente por permitir o avanço em campos como agricultura, pecuária, genética forense, transgenia e na microbiologia com a detecção e genotipagem de agentes patogênicos nos seres vivos (HEPP; NONOHAY, 2016). As primeiras técnicas de sequenciamento surgiram em 1975 por meio de Frederick Sanger com seus estudos baseados no método de terminação em cadeia e em 1977 por Allan Maxam e Walter Gilbert com a proposta do método químico de degradação de bases. Esses métodos são considerados a primeira geração de sequenciadores e permitiram, por exemplo, o sequenciamento do primeiro genoma bacteriano *(Haemophilus influenzae)* em 1995 (FLEISCHMANN et al., 1995; SCHADT; TURNER; KASARSKIS, 2010; MARIANO, 2015).

O método da terminação em cadeia acabou se popularizando e se tornou, durante três décadas, a principal técnica utilizada para o sequenciamento de ácidos nucléicos (III, 2007; MARTINS, 2013). O sequenciamento Sanger, como também ficou conhecido, se tornou tão relevante para comunidade científica que foi utilizado em um dos projetos mais ambiciosos da história: o Projeto Genoma Humano, sendo este um esforço internacional iniciado em 1989 para sequenciar os 3,1 bilhões de bases nitrogenadas do genoma humano, objetivo esse que foi concluído somente em abril de 2003 (GÓES; OLIVEIRA, 2014; LANDER et al., 2001; VENTER et al., 2001).

Embora utilizado em larga escala durante trinta anos, o sequenciamento Sanger apresentava alguns problemas, como a execução lenta, a incapacidade de gerar grande quantidade de dados e o seu alto custo de operação (MARIANO, 2015; BONETTA, 2006; POP, 2009). Segundo Mariano (2015), Schadt, Turner e Kasarskis (2010), o baixo rendimento e as insuficiências da tecnologia Sanger promoveram uma intensa procura por métodos que promovessem melhorias na técnica de sequenciar o DNA, principalmente em fatores como a redução de custos por base sequenciada e no tempo gasto para a conclusão do processo. Essa procura por métodos com maiores rendimentos acabou resultando no surgimento de uma segunda geração de sequenciadores, as chamadas tecnologias NGS *(Next Generation Sequencing)*.

A tecnologia NGS surgiu em meados do ano de 2005 com a empresa *454 Life Sciences* que desenvolveu uma nova técnica baseada em síntese, o pirosequenciamento. As vantagens promovidas por essa nova tecnologia foram enormes, destacando-se a capacidade de produção a baixo custo, e em grande quantidade, de dados biológicos sequenciados (CARVALHO; SILVA, 2010; RONAGHI et al., 1998; WETTERSTRAND, 2019). Além disso, essas plataformas possibilitaram uma compreensão mais detalhada de sequências genômicas inteiras e das informações nelas contidas, promovendo diversos avanços para a comunidade científica (SCHADT; TURNER; KASARSKIS, 2010).

O fato de serem técnicas baratas e com estratégias distintas do método Sanger acabou proporcionando um aumento na quantidade de projetos de sequenciamento publicados, principalmente de procariotos (MARIANO, 2015; HUSEMANN, 2011). Segundo Martins (2013), o volume de dados gerados pelas plataformas de segunda geração foi de duas a três ordens de magnitude maior do que os dados produzidos pela tecnologia Sanger, fato esse que gerou uma revolução na biologia.

Atualmente existem diversas plataformas de segunda geração disponíveis, sendo tal diversidade justificada pela variedade de métodos e técnicas para a execução desse processo. Dessa forma, alguns fatores acabam diferenciando uma plataforma da outra, como por exemplo: tempo de corrida – que nada mais é do que o tempo gasto para a conclusão do sequenciamento -, tamanho máximo dos fragmentos produzidos e a quantidade de fragmentos gerados por execução. Algumas plataformas, por exemplo, produzem *reads* (leituras ou fragmentos) curtos em grandes quantidades, enquanto outras produzem fragmentos longos em quantidades menores. Assim sendo, a análise desses fatores é essencial para a escolha da plataforma ideal para a execução do sequenciamento (OLIVEIRA, 2017).

As tecnologias de primeira e segunda geração - Sanger e NGS respectivamente - revolucionaram o campo da genômica e promoveram um número extraordinário de avanços científicos. Entretanto, o surgimento de elementos complexos durante o processo de sequenciamento fez com que tais tecnologias não fossem capazes de lidar com determinados fatores biológicos, criando, dessa maneira, um terreno fértil para inovações adicionais

nessa área. Assim, ao longo do tempo diversas técnicas foram desenvolvidas visando decifrar a sequência genética do DNA da maneira mais eficiente possível, o que resultou em plataformas de sequenciamento com metodologias totalmente distintas das demais, dando início a terceira geração de sequenciadores (SCHADT; TURNER; KASARSKIS, 2010).

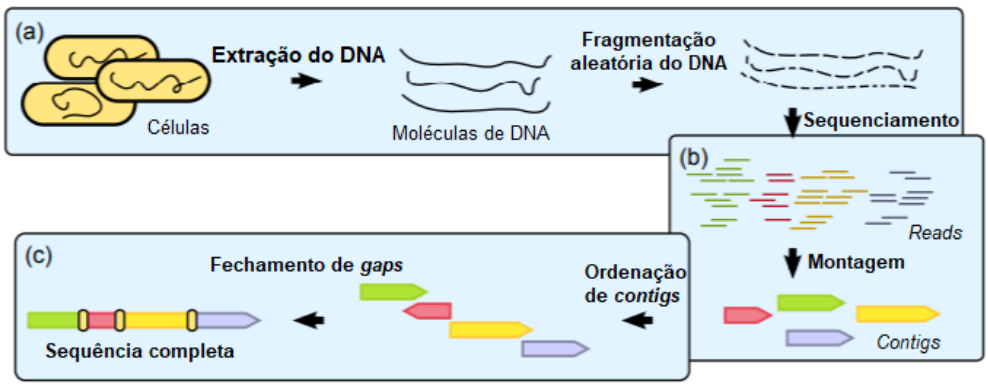
Enquanto as tecnologias NGS dependem da técnica de PCR *(Polymerase Chain Reaction)* para realizar o sequenciamento, a mais recente geração está sendo baseada em técnicas de sequenciamento de molécula única (SMS), que não utiliza PCR. Algumas das vantagens fornecidas pela terceira geração sobre as tecnologias anteriores são: maior produtividade, menor tempo de resposta, comprimento de leitura mais longo entre outros (SCHADT; TURNER; KASARSKIS, 2010).

2.2 Montagem de genomas

A etapa da montagem de genomas é um estágio posterior ao sequenciamento e tem por objetivo reordenar e colocar os fragmentos gerados na etapa anterior na ordem original, representando, assim, a molécula de DNA que compõe cada cromossomo da espécie em análise. Esse estágio é composto por *softwares* que utilizam de algoritmos computacionais para unir os fragmentos sequenciados de DNA de forma a obter uma representação do genoma original do organismo (MARTINS, 2013; MARIANO, 2015; WETTERSTRAND, 2019).

O método para efetuar a montagem de um genoma de modo completo é chamado por Husemann (2011) de *whole genome shotgun* (WGS) e descrito em etapas, sendo elas: extração e purificação do DNA, sequenciamento *shotgun*, montagem computacional e finalização do genoma. A figura abaixo mostra uma visão geral do processo descrito por Husemann.

Figura 1 – Etapas do método *whole genome shotgun sequencing* (WGS).



Fonte: Adaptado de HUSEMANN, 2011.

Inicialmente o DNA precisa ser extraído e purificado das células dos organismos para em seguida ser clonado, como evidenciado na Figura 1 (a). Logo depois ele é submetido a etapa de sequenciamento, mais especificamente o *shotgun*. Embora nos últimos anos as técnicas de sequenciamento tenham avançado consideravelmente, os métodos atuais ainda não são capazes de processar as bases de uma molécula de DNA em ordem e sequenciá-las em uma única etapa. A justificativa para isso é o tamanho dos genomas, que costumam ser grandes e acabam impossibilitando o seu processamento de uma única vez. Assim, para sequenciar um genoma, é preciso dividir aleatoriamente o seu DNA em pedaços menores e submeter cada fragmento resultante a reações químicas que permitam a identificação da ordem de suas bases. A fragmentação inicial aleatória fez com que essa abordagem ficasse conhecida como *"shotgun sequencing"*, ou sequenciamento de espingarda, fazendo alusão ao padrão aleatório dos projéteis de espingarda quando disparados contra um alvo. O produto da etapa de sequenciamento é chamado de *reads* ou leituras, que nada mais são do que fragmentos de DNA com a ordem de suas bases conhecidas, ou seja, sequenciadas (HUSEMANN, 2011; WETTERSTRAND, 2019).

Após o processo de sequenciamento, as leituras resultantes *(reads)* são destinadas ao estágio de montagem para serem reordenadas e representarem o genoma original (MILLER; KOREN; SUTTON, 2010). Para isso, as leituras são submetidas a um programa de computador que possui a finalidade de identificar regiões onde diferentes fragmentos possuem trechos coincidentes entre si. Em seguida, as partes coincidentes são sobrepostas e depois fundidas, permitindo então a união desses fragmentos e a realização da montagem (BROWN, 2002; MARTINS, 2013).

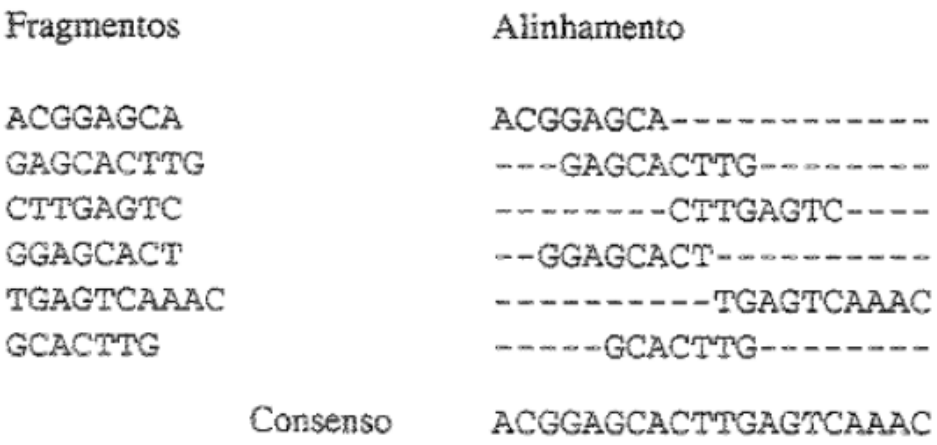
Uma variável importante desse estágio é a cobertura genômica, que pode ser definida como o número de vezes que uma determinada região do genoma é coberta por segmentos de leitura, contribuindo, consequentemente, para o processo de identificação da sequência de DNA na região considerada (MARTINS, 2013). As leituras geradas pelo sequenciamento são idealmente distribuídas de maneira uniforme por todo o genoma, promovendo, assim, uma montagem com uma cobertura alta o suficiente para determinar todas as bases do genoma (HUSEMANN, 2011).

De acordo com Husemann (2011), é nesse cenário que as atuais técnicas de sequenciamento se destacam, já que permitem gerar um grande número de leituras de maneira barata e, dessa maneira, fornecer a cobertura desejada para a etapa de montagem. No momento em que o genoma está "coberto" suficientemente pelos fragmentos, essas sobreposições são utilizadas para reconstruir trechos da sequência genômica, conhecidos como "sequência consenso" ou "sequência contígua" ou apenas *contigs*. O processo da produção de *contigs* a partir da sobreposição de um conjunto de *reads* - como evidenciado na Figura 2 abaixo - é referenciado como o estágio da montagem do genoma, sendo um exemplo típico de avanços na biologia obtidos por meio da bioinformática (HUSEMANN,

2011). Para melhor compreensão do processo, veja a seguir um exemplo.

Dado um conjunto de leituras sequenciadas, os programas de computador têm por objetivo construir um arranjo *(layout)* baseando-se nos trechos coincidentes entre os fragmentos. Para isso, os *softwares* realizam a sobreposição das regiões idênticas inserindo espaços (representado pelo caractere '-') a fim de que os segmentos sobrepostos fiquem alinhados. Desse modo, por meio da cobertura genômica formada, deduz-se então a sequência consenso (OKURA et al., 2002; SETUBAL; MEIDANIS, 1997).

Figura 2 – Caso ideal de montagem.



Fonte: Adaptado de SETUBAL; MEIDANIS, 1997.

Contudo, este exemplo apresenta somente um caso ideal do problema. Na prática, diversos fatores dificultam o processo de montagem, como a falta de cobertura genômica e o alto índice de repetições (OKURA et al., 2002; SETUBAL; MEIDANIS, 1997).

Existem duas maneiras de se realizar a montagem de genomas: por referência e *de novo*. O primeiro tipo diz respeito às montagens que utilizam uma sequência genômica para auxiliar no processo de alinhamento dos fragmentos de leitura, essa sequência é chamada então de "referência". Entretanto, na ausência de um genoma de referência, as leituras devem ser preparadas para a realização de uma montagem do tipo *de novo*. O termo *"de novo"* vem do latim e significa "desde o princípio" e representa métodos utilizados para determinar a sequência de DNA quando não há uma sequência genômica conhecida anteriormente para o uso como referência. Assim, somente os fragmentos sequenciados são utilizados no processo (MARTINS, 2013). No trabalho aqui desenvolvido, os montadores serão avaliados quanto a sua capacidade de realizar a montagem do tipo *de novo*.

2.2.1 Algoritmos computacionais

Os *softwares* que realizam a montagem do tipo *de novo* podem funcionar com base em três classes de algoritmos computacionais: algoritmo guloso *(greedy)*, OLC *(overlap-layout-consensus)* e grafo *de Bruijn* (DBG), em que cada um utiliza de estratégias próprias para lidar com os erros inerentes ao processo de montagem (MILLER; KOREN; SUTTON, 2010; OLIVEIRA, 2017).

2.2.1.1 Algoritmo Guloso/Ganancioso

O algoritmo guloso, também conhecido como algoritmo ganancioso ou *greedy*, baseia-se em uma sequência de etapas para solucionar um determinado problema. Cada uma dessas etapas possui um conjunto de escolhas, opções ou estratégias para satisfazer o objetivo final. Neste contexto, o algoritmo guloso aplica um critério de otimização e escolhe a estratégia que lhe parece melhor no momento. Isto é, ele busca fazer uma escolha localmente ideal na esperança de que essa escolha levará a uma solução globalmente ideal, mas nem sempre é possível chegar a uma solução ótima (ROCHA; DORINI, 2004; CORMEN et al., 2009; MORAIS, 2014). Esse algoritmo não se resume somente à montagem genômica, mas deve ser entendido como um conceito amplo de resolução de problemas.

O algoritmo guloso foi uma das primeiras estratégias utilizadas para a montagem *de novo*. Essa abordagem inicia o processo de montagem calculando alinhamentos (sobreposições) entre os fragmentos fornecidos. Por sobreposição, entende-se que o prefixo de uma leitura compartilha semelhanças com o sufixo de outra leitura. Dessa forma, o algoritmo inicia com os fragmentos que mais se sobrepõem e termina quando não houver mais fragmentos para mesclar. Esses alinhamentos são pontuados, normalmente tomando como base o comprimento da sobreposição e o nível de semelhança entre os pares de bases compartilhados (TAYLOR et al., 2012; POP, 2009).

No final, as leituras com as maiores pontuações de sobreposição são mescladas e transformadas em *contigs* (NARZISI; MISHRA, 2011).

2.2.1.2 *Overlap-layout-consensus* (OLC)

De acordo com Pop (2009), a classe de algoritmos OLC é dividida em três etapas. A primeira delas é a mesma da abordagem gulosa: as leituras são comparadas entre si com o objetivo de construir uma lista de sobreposição de pares dos fragmentos. Essas informações são usadas para construir um gráfico de sobreposição, no qual cada leitura é representada por um nó e uma borda é utilizada para conectar dois nós quando uma sobreposição é identificada entre as leituras correspondentes. A construção desse gráfico é, portanto, a segunda etapa requerida pela abordagem OLC. Em seguida, esse gráfico de sobreposição é analisado a fim de identificar caminhos que correspondam aos segmentos que estão sendo montados. O objetivo final é encontrar um único caminho que atravesse

todos os nós do gráfico exatamente uma única vez, resultando na reconstrução do genoma. Esse processo corresponde a terceira e última etapa da abordagem OLC.

2.2.1.3 Grafo *de Bruijn* (DBG)

O grafo *de Bruijn* é uma abordagem que não requer cálculos pesados de sobreposição. Ao invés disso, algoritmo calcula *k-mers* para construir o gráfico de montagem. Como as sobreposições não são calculadas explicitamente - como ocorre na abordagem OLC -, a montagem *de Bruijn* economiza uma quantidade substancial de tempo computacional (POP, 2009; TAYLOR et al., 2012).

Nesse tipo de abordagem, o algoritmo quebra os segmentos de leitura em partes menores antes da etapa de montagem, os quais em seguida são interligados por meio de um grafo para serem transformados em *contigs*. O tamanho dessas partes menores é definido pelo parâmetro *k-mer*, que nada mais é do que *substrings* de um determinado fragmento com tamanho K, sendo K um número inteiro positivo. O processo de formação de *contigs* é altamente sensível ao valor atribuído a esse parâmetro, já que qualquer valor de K - tanto os pequenos quanto os grandes - produz problemas complexos a serem solucionados pelas ferramentas de montagem. Ademais, é importante destacar que a estratégia da divisão dos segmentos em *k-mers* também é utilizada em certa medida pela abordagem OLC, principalmente na etapa de comparação par-a-par das leituras (MILLER; KOREN; SUTTON, 2010; MARIANO, 2015).

3 MATERIAIS E MÉTODOS

Neste capítulo será apresentado a metodologia e os materiais adotados para a confecção do estudo comparativo. A pesquisa aqui desenvolvida possui uma abordagem quantitativa e, quanto ao procedimento, bibliográfica. No que tange aos objetivos e a natureza respectivamente, o estudo pode ser classificado como explicativo e aplicado.

Outrossim, para alcançar os objetivos previamente descritos neste trabalho, foi necessário seguir impreterivelmente as seguintes etapas:

a. Revisão bibliográfica

Segundo Vergara (2006, p. 48), a revisão bibliográfica é "o estudo sistematizado desenvolvido com base em material publicado em livros, revistas, jornais, redes eletrônicas, isto é, material acessível ao público em geral".

Dessa forma, com o fito de compreender e descrever detalhadamente os processos que envolvem o sequenciamento de genomas, foram analisados diversos estudos - tanto nacionais quanto internacionais - na área da genômica computacional. Em nosso trabalho buscamos dar uma ênfase maior na etapa de montagem, explicitando os nuances e particularidades inerentes a esse processo.

b. Seleção dos programas de montagem

Como são muitos os programas de montagem disponíveis, estes foram submetidos a um processo de seleção e somente aqueles que atenderam aos critérios de escolha puderam participar do estudo comparativo.

c. Validação dos montadores com amostras biológicas

Com propósito de gerar material para análise, os montadores selecionados foram executados com dados biológicos provenientes de uma plataforma de sequenciamento de segunda geração.

d. Análise dos montadores

Após a etapa de validação, os dados produzidos pelos programas foram mensurados utilizando métricas de qualidade. Além de nos permitir aferir sobre a precisão dos resultados, esses parâmetros também possibilitaram a elaboração de análises sobre o comportamento e desempenho dos *softwares*.

3.1 Obtenção dos dados biológicos e plataforma de sequenciamento

Os dados biológicos utilizados para a validação das ferramentas foram obtidos no banco de dados do NCBI *(National Center for Biotechnology Information)*. O NCBI é uma instituição criada em 1988 destinada ao desenvolvimento de sistemas informacionais para auxiliar os estudos no campo da biologia molecular, armazenando e disponibilizando pesquisas, dados e uma série de informações aos cientistas do mundo todo. Para isso, esse centro mantém um banco de dados próprio chamado *GenBank* que abriga informações provenientes de projetos de sequenciamento feitos ao redor do mundo, sendo enviados ao instituto por meio de uma colaboração internacional (NCBI RESOURCE COORDINATORS, 2018). Todas as amostras selecionadas foram obtidas na base de dados SRA *(Sequence Read Archive)*, estão no formato *.fastq*, possuem extremidade única *(single-end)* e foram geradas por meio do método WGS. Além do mais, o grupo de pesquisa do orientador deste estudo já trabalhou com esses dados e, portanto, possui certa familiaridade com eles, fator esse que também influenciou a nossa escolha.

O *Sequence Read Archive* é um repositório internacional mantido pelo *International Nucleotide Sequence Database Collaboration* (INSDC) que armazena informações brutas de projetos de sequenciamento, ou seja, dados biológicos que não foram tratados ou aprimorados para a etapa de montagem (LEINONEN et al., 2010). Nos projetos de montagem de genomas, os arquivos a serem montados passam por uma etapa anterior, a de análise e tratamento dos dados. No estágio de análise, os organismos são submetidos a uma perícia para a averiguação da sua qualidade de sequenciamento. Assim, caso seja constatado um baixo índice nesse quesito, é realizado correções nos fragmentos com o propósito de melhorar a sua acurácia (estágio de tratamento) (OLIVEIRA, 2017).

Porém, a fim de submeter os programas de montagem as mesmas circunstâncias e testar ao máximo os seus módulos e recursos, nenhuma amostra utilizada neste estudo passou pela etapa de análise ou tratamento, sendo montadas como dados brutos. Além disso, a utilização dos arquivos SRA nas montagens nos permitiu compreender o modo como cada *software* lida com informações desse tipo e o nível de influência dos seus módulos e algoritmos na qualidade final dos resultados.

Os organismos foram sequenciados na plataforma *Ion Personal Genome Machine* (Ion PGM). O Ion PGM foi desenvolvido pela empresa *Ion Torrent Systems Inc.* e lançado no ano de 2010 como uma tecnologia baseada em semicondutores para realizar o sequenciamento. Esses semicondutores visam detectar mudanças de pH decorrentes da incorporação de nucleotídeos nas moléculas de DNA, em um processo conhecido como polimerização. Esse novo modo de sequenciar o DNA resultou em maior velocidade – produzindo leituras de 200 pb (pares de base) em até 2 horas -, maior estabilidade e uma redução nos custos de sequenciamento (LIU et al., 2012; FLUSBERG et al., 2010).

Na Tabela 1 abaixo é possível observar a nomenclatura das bibliotecas SRA utilizadas no estudo e os seus respectivos códigos de acesso no portal NCBI.

Tabela 1 – Bibliotecas SRA usadas no estudo.

Bibliotecas SRA	Número de bases (pb)	Código de acesso ao NCBI
Advenella kashmirensis WT001	252M	SRR788730
Clostridium autoethanogenum DSM 10061	99.5M	SRR1748018
Corynebacterium pilosum CIP103422	388.9M	SRR3312980
Mycobacterium ulcerans Mu F74	422M	ERR732677
Pedobacter sp. NL19	1.1G	SRR1769012
Staphylococcus aureus strain M34-B-1_11	328.5M	ERR493460

Fonte: Adaptado de OLIVEIRA, 2017.

3.2 Seleção dos programas de montagem

Para selecionar os montadores que iriam participar do estudo comparativo foi necessário, primeiramente, fazer um levantamento dos *softwares* de montagem atualmente disponíveis. Durante a sondagem foi possível constatar que nem todos os programas poderiam fazer parte do estudo, haja vista a incompatibilidade de alguns deles com os arquivos de entrada ou com a máquina disponível para os testes. Desse modo, foi preciso definir, por meio de um processo de seleção, quais montadores seriam testados.

Para isso, fizemos uma análise metódica de estudos e trabalhos publicados no campo da bioinformática com o fito de identificar quais programas, entre aqueles inicialmente catalogados, apresentavam os melhores resultados. Assim, a escolha dos montadores foi baseada nos seguintes critérios: desempenho satisfatório nos trabalhos analisados, arquitetura *open source*, *softwares* livres, compatibilidade com a máquina de testes e a capacidade de trabalhar com arquivos no formato *Ion Torrent*.

É importante ressaltar que todos os programas foram executados sob as mesmas circunstâncias de montagem, com os mesmos arquivos de entrada e com seus módulos padrões. Os montadores selecionados e as características inerentes a cada um - como o algoritmo no qual foi arquitetado e a versão utilizada - estão listados na Tabela 2 abaixo.

Tabela 2 – Comparativo dos montadores selecionados.

Montador	Algoritmo	Versão utilizada	Linguagem de programação	Sistema operacional
ABySS	*de Bruijn*	2.1.5	C++, C	Unix/Linux, Mac OS
Minia	*de Bruijn*	3.2.1	C++, Python	Unix/Linux, Mac OS
MIRA	OLC	4.9.5	C, C++	Unix/Linux, Mac OS
Ray	Híbrido	2.3.1	C++, Python	Unix/Linux
SOAPdenovo2	*de Bruijn*	2.04	C, C++	Unix/Linux, Mac OS, Windows
SPAdes	*de Bruijn*	3.12.0	C++, C, Python, Perl	Unix/Linux, Mac OS
Velvet	*de Bruijn*	1.2.10	C, C++, Perl	Unix/Linux

Fonte: Próprio autor.

3.3 Montadores selecionados

3.3.1 ABySS

O ABySS *(Assembly By Short Sequencing)* é uma ferramenta de montagem *de novo* destinada a leituras curtas e grandes genomas (JACKMAN et al., 2017).

A principal característica do ABySS consiste no fato de realizar uma representação distribuída do gráfico *de Bruijn*, permitindo, assim, o cálculo paralelo do algoritmo de montagem em uma rede de computadores comuns. Ademais, o montador funciona em dois estágios distintos de execução, no qual cada um possui módulos para aumentar a qualidade e eficácia da montagem, a exemplo da biblioteca *"sparsehash"* para reduzir o uso de memória e do módulo de remoção de erros nos dados de entrada (JACKMAN et al., 2017).

3.3.2 Minia

O Minia é um programa destinado a montagem de leituras sequenciadas de DNA com uma grande economia de memória. Para isso, o *software* baseia-se em uma representação sucinta do gráfico *de Bruijn* e possui uma estrutura destinada a remoção de falsos positivos críticos. Assim, os recursos computacionais necessários para executar o programa são significativamente menores se comparado aos outros montadores (CHIKHI; RIZK, 2013).

3.3.3 MIRA

O MIRA *(Mimicking Intelligent Read Assembly)* é um montador/mapeador de dados de sequência de DNA para projetos de genoma inteiro e EST / RNASeq. O montador foi projetado com a classe de algoritmos OLC *(overlap-layout-consensus)* e usa estratégias iterativas de multipasses. O programa possui diversos módulos e funções para aumentar a acurácia da montagem, a exemplo do editor automático para analisar e editar alinhamentos, rotinas para detectar e resolver possíveis desmontagens e vários outros (CHEVREUX et al., 2004). Além disso, o montador dispõe de parâmetros que permitem o usuário controlar todo e qualquer aspecto da montagem, moldando-a de acordo as suas necessidades.

Como dito no capítulo 2, a estratégia de dividir os segmentos de leitura em *k-mers* também é utilizada, em certa medida, pela abordagem OLC. No caso do montador MIRA, ele baseia-se na frequência *k-mer* para avaliar a cobertura das sequências de DNA durante o estágio de pré-montagem, para que assim ele possa localizar e marcar os segmentos que apresentam repetições. Desse modo, embora os valores de K não participem diretamente do algoritmo de montagem do MIRA, eles exercem um papel extremamente importante ao retirar do processo tais sequências, aumentando, assim, a qualidade final das leituras

consenso (CHEVREUX et al., 2004). Por fim, os índices de *k-mer* associados a este programa não foram descritos, já que, por padrão, o próprio montador atribui a ele um valor.

3.3.4 Ray

O Ray é um *software* dedicado a montagem de leituras curtas de DNA. Ele é escrito em C++ e pode ser executado paralelamente em vários computadores por meio do padrão MPI *(Message Passing Interface)*. O Ray foi desenvolvido utilizando o grafo *de Bruijn* (DBG) como estratégia principal de montagem. Porém, algumas de suas rotinas de execução se baseiam em fundamentos do algoritmo ganancioso/guloso, o que nos levou a classificá-lo como "híbrido". Para mais, da mesma forma que os outros programas, ele é constituído por diversos módulos independentes que almejam aumentar a eficácia da montagem (BOISVERT; LAVIOLETTE; CORBEIL, 2010).

3.3.5 SOAPdenovo2

O SOAPdenovo2 é o sucessor do SOAPdenovo e foi projetado especificamente para a execução de montagens do tipo *de novo* de grandes genomas, embora funcione bem em genomas menores como de bactérias e fungos (LUO et al., 2012).

O programa foi desenvolvido levando em consideração o cenário no qual estava inserido, o da produção em massa de leituras cada vez menores pelas tecnologias de sequenciamento de segunda geração. Dessa forma, o SOAPdenovo2 é especializado em sequências curtas de leitura para a realização da montagem. Além disso, como a maioria dos montadores atuais, o programa também utiliza a classe de algoritmos grafo *de Bruijn* como estratégia principal para a execução da montagem (LUO et al., 2012).

O *software* possui diversos módulos para aumentar a eficácia de suas execuções, a exemplo do *"sparse-pregraph"* que reduz consideravelmente o consumo computacional e produz informações adicionais para resolver repetições (LUO et al., 2012).

3.3.6 SPAdes

O SPAdes é um kit de ferramentas que possui diversos *pipelines* para a realização da montagem. O programa foi desenvolvido utilizando a classe de algoritmos grafo *de Bruijn*, sendo projetado inicialmente para genomas pequenos como de organismos bacterianos, não sendo, portanto, destinado a genomas maiores como dos mamíferos (BANKEVICH et al., 2012).

O SPAdes possui parâmetros e recursos específicos para trabalhar com diversos tipos de dados - incluindo *Ion Torrent* -, além de apresentar uma grande variedade de parâmetros para controlar a sua execução (BANKEVICH et al., 2012).

3.3.7 Velvet

O Velvet é um *software* que o utiliza o algoritmo DBG para a montagem e alinhamento de sequências genômicas muito curtas e com informações extremamente emparelhadas (ZERBINO; BIRNEY, 2008).

Da mesma forma que os demais montadores, o Velvet também possui módulos e recursos importantes para eficácia da montagem, como a sua capacidade de resolver repetições de forma gananciosa (ZERBINO; BIRNEY, 2008).

3.4 Métricas de avaliação

Após a execução dos programas, é necessário avaliar corretamente e de maneira imparcial a qualidade dos resultados obtidos, adquirindo, dessa maneira, informações importantes para a análise comparativa.

Para esse propósito, a ferramenta escolhida foi o *Quality Assessment Tool for Genome Assemblies* (QUAST), que utiliza métricas de qualidade para mensurar montagens genômicas (GUREVICH et al., 2013). As métricas escolhidas para o estudo foram: número total de *contigs* gerados, N50 e o tamanho do genoma montado. Como métrica adicional, também foi avaliado o tempo que cada montador levou para a conclusão do processo. Para mais, a definição de cada um desses parâmetros e a sua consequente interpretação sobre o desempenho dos programas será feito no capítulo 4. A versão utilizada do QUAST foi 5.0.2.

No que tange aos valores de *k-mer* escolhidos, percebemos que alguns deles não produziam bons resultados. Dessa forma, a fim de promover circunstâncias iguais para os montadores, todos os programas que utilizam a classe *de Bruijn* foram executados sob os mesmos valores de K, sendo eles: 21, 33, 55, 77, 99 e 127. Entretanto, somente os valores de *k-mer* que produziram os melhores índices foram analisados. Os demais resultados obtidos e os seus respectivos valores de K podem ser visualizados no Apêndice A.

3.5 Especificações de *hardware* e *software*

A máquina utilizada para a execução dos programas é um computador que possui, quanto aos requisitos técnicos, um processador i7 de sétima geração, 8 GB de memória RAM (DDR4), 2 GB de placa de vídeo dedicada (gDDR5) e um armazenamento interno de 1 TB. No que diz respeito ao sistema operacional, todas as aplicações foram feitas na versão 16.04 LTS da plataforma Ubuntu - construída sob o núcleo Linux.

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os resultados obtidos com a execução dos programas selecionados na Tabela 2. Os dados produzidos pelos montadores foram analisados utilizando os parâmetros de qualidade citados no capítulo 3, nos permitindo avaliar o desempenho e comportamento dos *softwares* quando executados com conjuntos de dados diversos.

Todos os resultados estão contidos na Tabela 3 abaixo.

Tabela 3 – Resultados do estudo comparativo. [1]

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução	*k-mer*
SRR788730	ABySS	10.213	1.858	5.207.023	31 min	77
	Minia	6.329	1.838	4.331.364	01 min	77
	MIRA	**201**	**154.638**	4.450.071	2h	-
	Ray	651	11.071	4.509.119	1h 21 min	33
	SOAPdenovo2	-	-	-	-	-
	SPAdes	315	122.212	4.453.758	34 min	77
	Velvet	3.906	633	2.426.199	02 min	127
SRR1748018	ABySS	2.557	5.492	4.417.496	02 min	55
	Minia	3.507	9.019	4.680.110	01 min	55
	MIRA	649	82.262	4.502.846	39 min	-
	Ray	2.924	11.697	4.679.370	38 min	33
	SOAPdenovo2	10.277	702	3.675.207	<01 min	55
	SPAdes	**350**	**224.364**	4.570.017	12 min	127
	Velvet	4.815	1.090	4.182.506	01 min	127
SRR3312980	ABySS	607	5.204	494.100	03 min	127
	Minia	536	19.293	512.284	01 min	127
	MIRA	2.037	10.346	1.175.962	38 min	-
	Ray	175	3.299	398.382	44 min	55
	SOAPdenovo2	353	5.131	426.631	<01 min	77
	SPAdes	**13**	**107.421**	405.969	11 min	127
	Velvet	3.280	254	943.961	01 min	127

[1] Os resultados superiores estão marcados em negrito. Os símbolos (-) indicam que o processo de montagem não foi concluído por erros ou falhas durante o processo.

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução	*k-mer*
ERR732677	ABySS	13.412	1.741	6.638.272	07 min	127
	Minia	5.208	3.760	4.958.218	01 min	55
	MIRA	3.016	4.960	5.698.708	3h 40 min	-
	Ray	2.141	4.787	5.161.169	1h 34 min	33
	SOAPdenovo2	11.137	979	4.874.703	<01 min	77
	SPAdes	**1.430**	**9.608**	5.276.962	31 min	127
	Velvet	6.411	595	3.452.199	02 min	127
SRR1769012	ABySS	35.892	1.727	7.444.015	16 min	33
	Minia	24.842	4.029	9.967.532	05 min	127
	MIRA	-	-	-	-	-
	Ray	631	19.685	6.628.898	5h 42 min	33
	SOAPdenovo2	12.432	1.875	7.091.089	05 min	77
	SPAdes	**125**	**242.633**	6.082.111	1h 18 min	127
	Velvet	110.853	115	6.132.058	60h 5 min	21
ERR493460	ABySS	7.761	9.696	3.516.915	06 min	77
	Minia	1.979	15.669	2.927.115	02 min	77
	MIRA	573	156.925	2.976.141	2h 15 min	-
	Ray	310	19.176	2.846.124	1h 22 min	33
	SOAPdenovo2	6.470	2.796	3.284.279	02 min	77
	SPAdes	**143**	**204.059**	2.796.238	16 min	127
	Velvet	4.531	649	2.465.771	01 min	127

Fonte: Próprio autor.

4.1 Número de *contigs*

O número total de *contigs* gerados na etapa de montagem está diretamente relacionado ao desempenho do montador durante o estágio de sobreposição das leituras sequenciadas de DNA (OLIVEIRA, 2017). Destarte, sabendo que os *contigs* são produtos desse processo de sobreposição, o programa que conseguir sobrepor o maior número de *reads* irá, consequentemente, produzir o menor número de sequências consenso ao final do processo. Portanto, os montadores que geraram os menores valores para essa métrica de qualidade foram aqueles que melhor desempenharam o módulo de sobreposição.

Ao analisar a tabela de resultados é possível constatar que o montador que obteve os menores índices foi o SPAdes. Os programas que mais se aproximaram dele foram o Ray e o MIRA, no qual o último até apresentou um resultado superior em uma montagem.

Os montadores Velvet, ABySS, SOAPdenovo2 e Minia foram os que apresentaram os maiores números de *contigs* produzidos. Na montagem da biblioteca ERR493460, por exemplo, enquanto o SPAdes e Ray produziram respectivamente 143 e 310 *contigs*, o ABySS, SOAPdenovo2, Velvet e Minia geraram 7.761, 6.470, 4.531 e 1.979 sequências consenso, respectivamente.

Ademais, diversos fatores influenciam o bom andamento dessa etapa, sendo uma delas as características dos dados de entrada. Variáveis como o tamanho dos fragmentos, por exemplo, pode favorecer determinadas arquiteturas de montagem em detrimentro de outras e vice-versa.

4.2 N50

Segundo Martins (2013), o N50 é um parâmetro utilizado para calcular o comprimento dos *contigs* montados, mais especificamente do menor comprimento de *contig* a partir do qual o somatório de todos as sequências consenso representa a metade do tamanho total do genoma. Esse índice é essencial para avaliação da montagem de genomas bacterianos, uma vez que é por meio dele que se infere a qualidade da montagem feita. Assim, quanto maior for o valor de N50, maior será a qualidade do genoma montado.

Analisando os dados obtidos, constatamos que o SPAdes foi o montador que gerou os maiores valores de N50 para 5 dos 6 arquivos SRA. A única exceção em seu desempenho foi na montagem da biblioteca SRR788730, que será discutido adiante. No restante dos casos o SPAdes obteve valores consideráveis nas montagens, a exemplo do N50 de 224.364 obtido na execução do arquivo SRR1748018, se distanciando do *software* MIRA que apresentou o segundo melhor resultado com o valor de 82.262. Dessa forma, devido aos seus valores superiores de N50, fica claro que o SPAdes foi o programa que apresentou as montagens com as maiores qualidades. Ademais, do mesmo modo que na seção anterior,

os montadores que mais se aproximaram do SPAdes foram o MIRA e o Ray, entretanto, com valores distantes do primeiro.

O montador que apresentou os menores valores de N50 foi o Velvet, gerando os menores índices para a maioria dos casos testados. Esse comportamento deve-se a diversos fatores, sobretudo pelo fato de os conjuntos de dados utilizados não serem ideais para a arquitetura de montagem do programa, que foi projetado exclusivamente para lidar com leituras curtas e emparelhadas *(paired-ends)*.

4.3 Tamanho do genoma *(total length)*

O tamanho do genoma nada mais é do que o número total de bases existentes no organismo montado (GUREVICH et al., 2013). Destarte, por meio da análise dos resultados, é possível constatar que os valores não variaram em grandes proporções, salvo algumas exceções. Estas exceções - em sua maioria - encontram-se nos montadores Velvet, ABySS, SOAPdenovo2 e Minia, uma vez que geraram valores de *total length* com alguns desvios ao padrão estabelecido pelos outros *softwares*. Além do mais, tendo em vista que estes foram os programas que apresentaram os maiores valores no estágio de formação das leituras consenso, podemos pressupor que o baixo desempenho dos montadores nesse estágio pode ter influenciado no tamanho final do genoma montado.

Na montagem da biblioteca SRR788730, por exemplo, o programa Velvet apresentou um *total lenght* de 2.426.199 pb, se distanciando dos outros montadores que geraram mais de 4 milhões de pares de base. O mesmo aconteceu com o montador Minia ao executar o arquivo SRR1769012, gerando mais de 9 milhões de pares de base enquanto os outros *softwares* apresentaram valores na faixa dos 6 milhões. Do mesmo modo, na montagem da biblioteca SRR1748018, o programa SOAPdenovo2 foi o único montador a gerar um *total length* menor que 4 milhões de pares de base. Por fim, o ABySS também apresentou alguns valores irregulares, sendo a única ferramenta a gerar um tamanho de genoma maior que 5 milhões de pares de base para a biblioteca ERR732677.

O tamanho do genoma é uma métrica importante para avaliarmos o desempenho dos montadores, entretanto, não é primordial, já que alguns programas podem se equivocar no momento de gerar *contigs* (MARIANO, 2015).

4.4 Tempo de execução

O tempo de execução de uma montagem não é um fator que exerce influência sobre a qualidade final dos resultados. Contudo, é uma métrica relevante para avaliarmos o comportamento dos programas e compará-los entre si.

De modo geral, os *softwares* que executaram o processo de montagem em menos tempo foram o SOAPdenovo2, Minia, Velvet e ABySS. Todas essas ferramentas possuem recursos para reduzir o tempo de corrida e o uso de memória RAM durante a sua execução. Entretanto, nenhum desses artifícios proporcionaram aumento na qualidade dos resultados, visto que estes foram os programas que geraram os menores índices nas métricas de qualidade. Assim, podemos afirmar que os tempos apresentados por eles não foram somente consequência dos recursos presentes em sua arquitetura, mas também de possíveis erros na execução da montagem. Ademais, a ausência de módulos mais precisos para analisar os dados de entrada também pode ter reduzido o tempo de corrida, uma vez que os montadores não precisaram investir recursos computacionais para executar tais processos.

Por outro lado, o *software* que mais requisitou tempo foi o MIRA. Esse comportamento do programa pode ser justificado pela classe de algoritmos presente em sua arquitetura, o OLC *(overlap-layout-consensus)*, que em algumas situações pode apresentar um alto custo operacional (OLIVEIRA, 2017). O custo operacional pode ser definido como o conjunto de valores associados ao tempo de execução e a alocação de memória RAM *(Random Access Memory)* necessários para a realização de uma montagem. O tempo computacional é influenciado por dois fatores: complexidade do conjunto de dados e estratégia adotada pelo programa para a execução do processo (ZHANG et al., 2011). Dessa maneira, podemos afirmar que a abordagem utilizada pelo MIRA demandou maiores recursos de processamento e promoveu, consequentemente, um tempo maior para a conclusão da tarefa. Além disso, dependendo do conjunto de dados utilizado como entrada, trabalhar com OLC pode se tornar um problema, principalmente devido à complexidade de manusear os vértices no gráfico de sobreposição durante a construção do *layout* de montagem (MILLER; KOREN; SUTTON, 2010).

Para mais, os resultados também evidenciaram que, no geral, os montadores *de Bruijn* apresentaram tempos de execução menores se comparados aos programas que utilizaram outras estratégias de montagem, a exemplo do MIRA (OLC) e Ray (híbrido). Embora este último tenha sido desenvolvido com base no grafo *de Bruijn*, ele também foi projetado sob algumas rotinas do algoritmo ganancioso/guloso. Assim sendo, podemos afirmar que a estratégia híbrida adotada pelo Ray resultou em um tempo maior para a conclusão das montagens quando comparado aos programas que foram desenvolvidos unicamente sobre o grafo *de Bruijn*, apresentando valores próximos e até mesmo superiores ao MIRA.

4.5 Desempenho

Para a confecção da Tabela 3 e análise dos resultados apresentados nas seções anteriores, foram executadas cerca de 200 montagens.

Segundo Mariano (2015, p.63), "considera-se o melhor resultado para uma montagem *de novo*, o teste que apresentar: o menor número de *contigs* formados e maior valor de N50". Dessa forma, mediante a nossa investigação, foi possível concluir que o montador que apresentou os resultados mais satisfatórios foi o SPAdes. Isso deve-se a diversos fatores, como a classe de algoritmos no qual foi escrito - grafo *de Bruijn* -, a existência de parâmetros distintos para controlar a sua execução e os módulos presentes em sua arquitetura. Entre esses módulos está o *IonHammer*, um recurso nativo destinado a correção de erros de leituras do tipo *Ion Torrent*, sendo executado automaticamente no início da montagem.

A única exceção no desempenho do programa foi na execução da biblioteca SRR788730, no qual o módulo *IonHammer* apresentou um erro no processamento do arquivo de entrada. Portanto, a única montagem em que o módulo de correção não foi executado foi a mesma na qual o *software* obteve o seu menor desempenho. Assim, ficou perceptível a importância desse recurso para a qualidade dos resultados, se revelando como a principal estratégia adotada pelo SPAdes para lidar com os dados brutos de sequenciamento.

Para mais, ao comparar os resultados obtidos pelos outros *softwares*, é possível constatar que o montador que mais se aproximou dos valores do SPAdes foi o MIRA. Contudo, embora tenha apresentado resultados interessantes para análise, o MIRA foi projetado utilizando a classe de algoritmos OLC que, como já visto anteriormente, pode acarretar em um alto custo operacional, demandando ainda mais recursos da memória RAM e um tempo maior de execução. Isso pode ser comprovado analisando os tempos de corrida gerados pelo programa, apresentando, na maioria dos casos, os maiores valores. Na montagem da biblioteca ERR732677, por exemplo, o SPAdes completou o processo em 31 minutos, diferente do MIRA que precisou de 3 horas e 40 minutos para a concluí-lo.

Além disso, o MIRA foi o único programa que não conseguiu concluir a montagem da biblioteca SRR1769012. Com o fito encontrar os motivos que impediram a execução do processo, analisamos o arquivo de registro do *software* e encontramos um erro antes etapa de montagem, mais especificamente no estágio de validação das leituras. Assim, a hipótese mais provável para a não execução do programa deve-se as informações brutas de sequenciamento que este, por sua vez, não conseguiu processar. Dessa forma, mesmo possuindo nativamente recursos para tratar dados do tipo *Ion Torrent*, o MIRA não foi capaz de montar um dos seis organismos.

Ademais, é importante analisarmos o comportamento anômalo apresentado por esse

montador durante a execução da biblioteca SRR3312980, na qual ele obteve o seu menor desempenho. As condições do arquivo de entrada - contendo uma cobertura genômica superior ao recomendado pelo *software* - acabou promovendo circunstâncias desfavoráveis para a sua execução, levando-o a produzir baixos valores nas métricas de qualidade. Para mais, ao observarmos os resultados obtidos pelos outros montadores, fica perceptível que não foi somente o MIRA que apresentou dificuldades para realizar a montagem desse organismo, já que a maioria dos *softwares* também apresentaram um desempenho insatisfatório.

Entre os 7 programas testados, o Ray foi o que mais apresentou oscilações e até mesmo incoerência nos resultados, visto que o programa obteve bons índices no número de *contigs* gerados e não conseguiu refletir esses resultados nos valores de N50, produzindo números bem abaixo de outros programas como SPAdes e MIRA. Na montagem da biblioteca ERR493460, por exemplo, embora o Ray tenha produzido 310 *contigs* - valor inferior somente ao do SPAdes - o seu índice de N50 foi de 19.176, sendo um dos mais baixos.

Ao fim do estudo também foi possível constatar que os montadores que apresentaram os menores índices nas métricas de qualidade foram o Velvet, SOAPdenovo2, ABySS e Minia. São diversos os motivos que contribuíram para os índices negativos, sendo o principal deles o baixo desempenho na etapa de formação das leituras consenso, justificando, assim, o grande número de *contigs* produzidos. Além disso, levando em consideração as condições dos arquivos entrada - possuindo regiões de cobertura desigual e com grandes índices de repetição, características comuns dos arquivos SRA -, muitos *softwares* utilizam de recursos próprios para lidar com esses fatores e aumentar a eficácia de sua montagem. Portanto, podemos afirmar que a ausência de módulos mais eficientes para lidar com as características complexas provenientes do processo de sequenciamento pode ter contribuído consideravelmente para os resultados insatisfatórios.

Em adição, é importante ressaltar que, do mesmo modo que o MIRA, o SOAPdenovo2 também não foi capaz de completar a montagem de um dos arquivos SRA, mais especificamente da biblioteca SRR788730, executando o processo por mais de uma semana e mesmo assim não sendo capaz de concluí-lo. Infelizmente, o montador não gerou qualquer registro sobre o processo feito, impossibilitando uma análise apurada sobre a sua não execução. Contudo, sabendo que o SPAdes apresentou dificuldades para realizar o pré-processamento dessa mesma biblioteca, podemos supor que o mesmo aconteceu com o SOAPdenovo2, porém, diferente do primeiro, esse fator impediu por completo a montagem do genoma.

No que se refere ao parâmetro *k-mer*, percebemos que os valores escolhidos exerceram forte influência sobre os montadores. Alguns programas se mostraram mais eficientes com altos valores de K - como 127 - o que aconteceu com os *softwares* SPAdes e Velvet. Em

contrapartida, o montador Ray apresentou um desempenho superior ao ser executado com valores menores para esse parâmetro, especialmente o 33 e 55. O *software* SOAPdenovo2, por outro lado, obteve melhores resultados com o valor 77, entretanto, o programa não foi capaz de executar nenhuma montagem com os valores 99 e 127. Os programas Minia e ABySS foram os mais equilibrados, apresentando resultados positivos tanto para os altos valores quanto para os pequenos. Dessa forma, podemos concluir que os índices de K variam de acordo o montador utilizado, não havendo, portanto, métodos que determinem com exatidão o valor ideal, sendo necessário escolhê-lo manualmente.

Por fim, é importante destacar que o processo de montagem é altamente dependente do poder de processamento da máquina disponível para os testes e dos dados fornecidos como entrada. Desse modo, os índices aqui apresentados não podem ser usados para descrever todos os cenários relacionados a montagem de genomas, pois caso os *softwares* fossem testados em outras máquinas ou com formatos de dados diferentes, os resultados não seriam os mesmos. Assim sendo, o SPAdes foi definido como o montador de desempenho mais satisfatório mediante as configurações do computador utilizado e, principalmente, das leituras do tipo *Ion Torrent*.

4.6 Usabilidade e implementação

Nosso objetivo com essa seção é compartilhar a nossa experiência ao instalar e executar tais ferramentas. Contudo, compreendemos que essas informações são subjetivas e que dependem das configurações da máquina e do perfil do utilizador. Entretanto, acreditamos que o seu compartilhamento poderá fornecer aos usuários leigos percepções gerais acerca do nível de dificuldade para implementar e executar tais programas.

De modo geral, os montadores presentes neste estudo possuem manuais com informações precisas sobre o processo de instalação. Além disso, a existência de binários pré-compilados em diversos repositórios evitou um contato direto com o código fonte dos *softwares*, facilitando a implementação. Todavia, a maioria desses montadores exigem como pré-requisito a instalação de outros programas para serem executados o que, em certas circunstâncias, pode gerar dificuldades. Em alguns casos, as dependências pré-instaladas no sistema operacional possuem versões antigas que precisam ser atualizadas para o seu correto funcionamento, e em outros elas podem apresentar incompatibilidade com o sistema ou com os pré-requisitos nele instalados, exigindo, portanto, atenção durante todo o processo.

É importante salientar que alguns programas tiveram suas configurações padrões alteradas durante a instalação, como foi o caso dos montadores Velvet e Ray que possuíam um tamanho limitado para o parâmetro *k-mer*, com 31 sendo o valor máximo. Assim, estes tiveram seus scripts de configurações modificados a fim de aceitarem tamanhos maiores

para se encaixarem na proposta do trabalho. Os detalhes acerca da instalação e execução desses programas estão disponíveis no Anexo A.

No que tange à usabilidade dos *softwares* de montagem, este depende do fluxo de trabalho adotado por cada programa, sendo possível dividi-los em dois grupos. O primeiro refere-se aqueles que funcionam via terminal (linha de comando) e o segundo é composto pelos montadores que utilizam os arquivos de configuração. Levando em consideração os programas testados neste estudo, o primeiro grupo é formado por SPAdes, ABySS, Velvet, Minia e Ray, enquanto o segundo é constituído pelo MIRA. Nesse quesito, o SOAPdenovo2 pode ser considerado um programa híbrido, já que seu funcionamento depende tanto de recursos da linha de comando quanto do arquivo de configuração. Para mais, nenhum desses programas possuem interface gráfica, o que pode representar um obstáculo para pessoas com pouco conhecimento em informática.

Na maior parte dos casos, os montadores que funcionam via terminal foram mais fáceis de executar, principalmente devido à disposição sequencial das informações de montagem, o que demandou menos tempo para a compreensão do fluxo de trabalho. Contudo, caso o usuário utilize muitos parâmetros ou decida por montagens mais detalhadas, a leitura e compreensão pelo terminal pode se tornar complexa devido ao grande volume de informações.

Sob esse aspecto, os montadores baseados nos arquivos de configuração foram mais eficientes em organizar tais informações. Entretanto, após diversas pesquisas em fóruns relacionados à bioinformática, encontramos relatos de usuários com dificuldades em executar programas desse tipo, sobretudo para entender a disposição dos seus parâmetros. Isso talvez esteja atrelado a ausência de padronização desses documentos, o que fez com que cada *software* adotasse um modelo único.

Todavia, esse comportamento não é uma regra. A utilização desses modelos depende unicamente do modo como os desenvolvedores o implementam na ferramenta. Portanto, ambos representam poderosas metodologias de montagem e, caso sejam dominados, vão ser de grande valia para o usuário.

5 Conclusão

A montagem e o sequenciamento de genomas bacterianos é uma área em ascensão e de grande relevância para a comunidade científica, sobretudo por promover o avanço em campos como a medicina, agricultura, pecuária e biologia molecular.

Entretanto, a produção crescente de ferramentas computacionais tornou o desenvolvimento de pesquisas na área algo substancialmente mais complexo e oneroso, pois, como consequência, muitos profissionais ficaram inseguros durante a escolha de uma ferramenta de montagem para a sua pesquisa, exigindo um maior investimento em recursos financeiros e disponibilidade de tempo.

Diante o exposto, o principal objetivo deste trabalho foi avaliar, por meio de um estudo comparativo, diferentes *softwares* computacionais para montagem *de novo* de genomas bacterianos. Para tal, estes foram submetidos a um processo de seleção e precisavam atender a alguns pré-requisitos, como o fato de serem gratuitos e possuírem uma arquitetura *open source*. Os montadores selecionados foram: ABySS, Minia, MIRA, Ray, SOAPdenovo2, SPAdes e Velvet, sendo executados com arquivos brutos de projetos de sequenciamento provenientes da plataforma *Ion Personal Genome Machine* (Ion PGM).

Os programas foram avaliados sob algumas perspectivas, compreendendo o seu desempenho e o nível geral de implementação e usabilidade. Para a interpretação dos resultados alcançados nós utilizamos a ferramenta QUAST, baseando-se nas seguintes métricas de qualidade: N50, número de *contigs* e *total length*.

Mediante a investigação aqui desenvolvida, foi possível concluir que a ferramenta que apresentou os resultados mais satisfatórios foi o SPAdes. Além disso, o estudo também revelou a influência que os recursos e estratégias de montagem exerceram sobre o desempenho de cada programa. Montadores como SPAdes e MIRA, por exemplo, que possuem módulos eficientes para lidar com dados brutos sequenciamento, geraram resultados superiores aos que não possuem tais recursos ou que os executaram de maneira insatisfatória, como aconteceu com os programas Velvet, SOAPdenovo2, ABySS e Minia.

Para mais, não foram somente os recursos de montagem que se mostraram relevantes para desempenho dos programas, mas também a classe de algoritmos nos quais foram desenvolvidos. Em suma, os montadores *de Bruijn* se mostraram mais eficientes no gerenciamento de recursos computacionais quando comparados aos programas que utilizaram outras estratégias. Por fim, no que tange ao nível de usabilidade e implementação dos *softwares*, averiguamos que aqueles que funcionam via terminal se mostraram intuitivamente mais eficientes que os arquivos de configuração. Em contrapartida, este último se destacou na organização das variáveis de montagem.

Dos resultados alcançados conclui-se que o desenvolvimento de módulos mais eficientes para o tratamento dos dados de entrada se revela uma estratégia importante para reduzir o custo operacional dos programas e promover resultados com maiores qualidades. Além disso, acreditamos que o desenvolvimento de *softwares* híbridos que busquem unir os pontos positivos das ferramentas aqui analisadas é uma outra estratégia com futuro promissor, uma vez que pode aumentar consideravelmente a eficácia das execuções. Por último, destacamos também a necessidade de ferramentas com interfaces mais amigáveis para uma maior democratização do seu uso.

Ademais, como em toda pesquisa científica, este estudo também possui suas limitações. Entre elas, podemos citar as restrições de recursos computacionais e amostras biológicas, que restringiu o número de programas a serem testados e dificultou o pleno funcionamento de alguns deles.

Em trabalhos futuros almejamos superar essas dificuldades e pesquisar sobre outras plataformas de sequenciamento, sobretudo aquelas com estratégias híbridas de execução, avaliando o modo como estas influenciam a qualidade final da montagem. Em adição, almejamos também incrementar outras métricas de qualidade e investigar as novas ferramentas que estão sendo desenvolvidas, incluindo os recursos e módulos atuais de montagem.

O trabalho aqui desenvolvido será de suma importância para pesquisadores e cientistas da área, principalmente por reduzir o tempo aplicado em pesquisas e análises para encontrar o montador adequado aos seus projetos. Além disso, as informações contidas neste estudo podem ser usadas para o aprimoramento das ferramentas e técnicas relacionadas à montagem de genomas bacterianos, revelando-se como uma importante fonte de informações para pesquisadores no campo da biologia computacional.

Referências

ALVES, S. M. A bioinformática e sua importância para a biologia molecular. *Revista Brasileira de Educação e Saúde*, v. 3, n. 4, p. 18–25, 2013.

ANDRADE, M. A. B. S. de; CALDEIRA, A. M. de A. O modelo de dna e a biologia molecular: inserção histórica para o ensino de biologia. *Filosofia e historia da biologia*, Associação Brasileira de Filosofia e História da Biologia-ABFHiB, v. 4, n. 1, p. 139–165, 2009.

BANKEVICH, A. et al. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 19, n. 5, p. 455–477, 2012.

BOISVERT, S.; LAVIOLETTE, F.; CORBEIL, J. Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *Journal of computational biology*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 17, n. 11, p. 1519–1533, 2010.

BONETTA, L. Genome sequencing in the fast lane. *Nature Methods*, Nature Publishing Group, v. 3, n. 2, p. 141–147, 2006.

BROWN, T. A. Sequencing genomes. In: *Genomes. 2nd edition*. [S.l.]: Wiley-Liss, 2002.

CARVALHO, M. C. d. C. G. d.; SILVA, D. C. G. d. Sequenciamento de dna de nova geração e suas aplicações na genômica de plantas. *Ciência Rural*, SciELO Brasil, v. 40, n. 3, p. 735–744, 2010.

CERDEIRA, L. T. et al. Rapid hybrid de novo assembly of a microbial genome using only short reads: Corynebacterium pseudotuberculosis i19 as a case study. *Journal of Microbiological Methods*, Elsevier, v. 86, n. 2, p. 218–223, 2011.

CHEVREUX, B. et al. Using the miraest assembler for reliable and automated mrna transcript assembly and snp detection in sequenced ests. *Genome research*, Cold Spring Harbor Lab, v. 14, n. 6, p. 1147–1159, 2004.

CHIKHI, R.; RIZK, G. Space-efficient and exact de bruijn graph representation based on a bloom filter. *Algorithms for Molecular Biology*, Springer, v. 8, n. 1, p. 22, 2013.

CORMEN, T. H. et al. *Introduction to algorithms*. [S.l.]: MIT press, 2009.

DIDELOT, X. et al. Transforming clinical microbiology with bacterial genome sequencing. *Nature Reviews Genetics*, Nature Publishing Group, v. 13, n. 9, p. 601–612, 2012.

FLEISCHMANN, R. D. et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, American Association for the Advancement of Science, v. 269, n. 5223, p. 496–512, 1995.

FLUSBERG, B. A. et al. Direct detection of dna methylation during single-molecule, real-time sequencing. *Nature methods*, Nature Publishing Group, v. 7, n. 6, p. 461, 2010.

GÓES, A. C. d. S.; OLIVEIRA, B. V. X. d. Projeto genoma humano: um retrato da construção do conhecimento científico sob a ótica da revista ciência hoje. *Ciência & Educação (Bauru)*, SciELO Brasil, v. 20, n. 3, p. 561–577, 2014.

GUREVICH, A. et al. Quast: quality assessment tool for genome assemblies. *Bioinformatics*, Oxford University Press, v. 29, n. 8, p. 1072–1075, 2013.

HEPP, D.; NONOHAY, J. S. de. A importância das técnicas e análises de dna. *ScientiaTec, Porto Alegre*, v. 3, n. 2, p. 114–124, 2016.

HUSEMANN, P. Bioinformatic approaches for genome finishing. 2011.

III, C. A. H. Dna sequencing: bench to bedside and beyond. *Nucleic acids research*, Oxford University Press, v. 35, n. 18, p. 6227–6237, 2007.

JACKMAN, S. D. et al. Abyss 2.0: resource-efficient assembly of large genomes using a bloom filter. *Genome research*, Cold Spring Harbor Lab, v. 27, n. 5, p. 768–777, 2017.

KIRCHER, M.; KELSO, J. High-throughput dna sequencing–concepts and limitations. *Bioessays*, Wiley Online Library, v. 32, n. 6, p. 524–536, 2010.

LANDER, E. S. et al. Initial sequencing and analysis of the human genome. Macmillan Publishers Ltd., 2001.

LEINONEN, R. et al. The sequence read archive. *Nucleic acids research*, Oxford University Press, v. 39, n. suppl_1, p. D19–D21, 2010.

LIMA, A. C. Ancoragem de genomas incompletos em genomas completos. 2007.

LIU, L. et al. Comparison of next-generation sequencing systems. *BioMed research international*, Hindawi Publishing Corporation, v. 2012, 2012.

LUO, R. et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, Oxford University Press, v. 1, n. 1, p. 2047–217X, 2012.

MARIANO, D. C. B. Simba: uma ferramenta web para gerenciamento de montagens de genomas bacterianos. Universidade Federal de Minas Gerais, 2015.

MARTINS, A. M. Sequenciamento de dna, montagem de novo do genoma e desenvolvimento de marcadores microssatélites, indels e snps para uso em análise genética de brachiaria ruziziensis. 2013.

MILLER, J. R.; KOREN, S.; SUTTON, G. Assembly algorithms for next-generation sequencing data. *Genomics*, Elsevier, v. 95, n. 6, p. 315–327, 2010.

MODA, T. L. *Desenvolvimento de modelos in silico de propriedades de ADME para a triagem de novos candidatos a fármacos*. Tese (Doutorado) — Universidade de São Paulo, 2007.

MOORE, P. Francis crick dies. *Genome Biology*, BioMed Central, v. 5, n. 1, p. 1–5, 2004.

MORAIS, C. M. Algoritmo guloso. 2014.

NARZISI, G.; MISHRA, B. Comparing de novo genome assembly: the long and short of it. *PloS one*, Public Library of Science, v. 6, n. 4, p. e19175, 2011.

NCBI RESOURCE COORDINATORS. Database resources of the national center for biotechnology information. *Nucleic acids research*, Oxford University Press, v. 46, n. D1, p. D8–D13, 2018.

OKURA, V. K. et al. Bioinformática de projetos genoma de bactérias. [sn], 2002.

OLIVEIRA, M. B. d. Gatool-genome assembly tool: uma ferramenta web para montagem de genomas bacterianos. Universidade Estadual de Feira de Santana, 2017.

OTTO, T. D. et al. A plataforma pdtis de bioinformática: da seqüência à função. *Revista Eletrônica de Comunicação, Informação e Inovação em Saúde*, v. 1, n. 2, 2007.

POP, M. Genome assembly reborn: recent computational challenges. *Briefings in bioinformatics*, Oxford University Press, v. 10, n. 4, p. 354–366, 2009.

RAMB. A descoberta do DNA e o projeto genoma. *Revista da Associação Médica Brasileira*, scielo, v. 51, p. 1 – 1, 02 2005. ISSN 0104-4230. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-42302005000100001&nrm=iso>.

RAMOS, R. T. J. et al. Tips and tricks for the assembly of a c orynebacterium pseudotuberculosis genome using a semiconductor sequencer. *Microbial Biotechnology*, Wiley Online Library, v. 6, n. 2, p. 150–156, 2013.

ROCHA, A.; DORINI, L. B. Algoritmos gulosos: definições e aplicações. *Campinas, SP*, v. 53, 2004.

RONAGHI, M. et al. A sequencing method based on real-time pyrophosphate. *Science*, v. 281, n. 5375, p. 363–365, 1998.

SAUNDERS, K. C. Automation and robotics in adme screening. *Drug Discovery Today: Technologies*, Elsevier, v. 1, n. 4, p. 373–380, 2004.

SCHADT, E. E.; TURNER, S.; KASARSKIS, A. A window into third-generation sequencing. *Human molecular genetics*, Oxford University Press, v. 19, n. R2, p. R227–R240, 2010.

SCHEID, N. M. J.; DELIZOICOV, D.; FERRARI, N. A proposição do modelo de dna: um exemplo de como a história da ciência pode contribuir para o ensino de genética. *Encontro Nacional de Pesquisa em Educação em Ciências*, v. 4, 2003.

SETUBAL, J. C.; MEIDANIS. *Introduction to computational molecular biology*. [S.l.]: PWS Pub. Boston, 1997.

SOLÓRZANO, L. P. et al. Impacto de la bioinformática en las ciencias biomédicas. *Acimed*, 2000, Editorial Ciencias Médicas, v. 11, n. 4, p. 0–0, 2003.

TAYLOR, D. L. et al. *PHAST (Phage Assembly Suite and Tutorial): A Web-based Genome Assembly Teaching Tool*. Tese (Doutorado) — Davidson College, 2012.

VALLE, M. In silico – uma alternativa viável aos experimentos in vivo? 2019.

VENTER, J. C. et al. The sequence of the human genome. *science*, American Association for the Advancement of Science, v. 291, n. 5507, p. 1304–1351, 2001.

VERGARA, S. C. Projetos e relatórios de pesquisa. *São Paulo: Atlas*, 2006.

VERLI, H. Bioinformática: da biologia à flexibilidade molecular. Sociedade Brasileira de Bioquímica e Biologia Molecular, 2014.

WETTERSTRAND, K. The cost of sequencing a human genome. *NHGRI https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost*, 2019.

ZERBINO, D. R.; BIRNEY, E. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, Cold Spring Harbor Lab, v. 18, n. 5, p. 821–829, 2008.

ZHANG, W. et al. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PloS one*, Public Library of Science, v. 6, n. 3, p. e17915, 2011.

Apêndices

APÊNDICE A – Resultados das montagens de acordo ao parâmetro *k-mer*

Tabela 4 – *k-mer* 21

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	50.634	1.634	5.613.808	11 min
	Minia	212.208	1.857	9.094.801	02 min
	Ray	1.541	4.517	4.678.803	1h 35 min
	SOAPdenovo2	-	-	-	-
	SPAdes	19.462	11.575	4.882.544	07 min
	Velvet	440.405	128	29.952.166	16h 10min
SRR1748018	ABySS	10.709	2.356	4.512.167	02 min
	Minia	12.531	2.350	4.811.441	01 min
	Ray	4.381	7.934	4.797.240	43 min
	SOAPdenovo2	64.891	776	8.155.267	<01 min
	SPAdes	6.356	4.304	4.763.231	12 min
	Velvet	100.258	166	8.673.072	01 min
SRR3312980	ABySS	1.587	2.041	441.409	01 min
	Minia	5.631	1.309	1.132.793	01 min
	Ray	5.252	2.709	1.302.970	40 min
	SOAPdenovo2	37.752	116	1.763.950	01 min
	SPAdes	598	1.542	648.199	11 min
	Velvet	38.243	128	2.703.775	02 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	28.988	1.518	5.639.162	06 min
	Minia	16.942	2.212	5.422.524	01 min
	Ray	2.385	4.816	5.167.968	43 min
	SOAPdenovo2	86.724	758	7.701.849	06 min
	SPAdes	3.813	4.411	5.151.434	31 min
	Velvet	111.874	143	6.996.161	07 min
SRR1769012	ABySS	39.508	1.522	6.958.873	13 min
	Minia	46.098	1.852	7.072.202	05 min
	Ray	1.048	11.647	6.130.293	5h 35 min
	SOAPdenovo2	280.462	563	14.484.809	1h 5 min
	SPAdes	2.538	13.429	6.025.492	1h
	Velvet	110.853	115	6.132.058	60h 5 min
ERR493460	ABySS	23.376	3.214	3.352.328	04 min
	Minia	17.371	3.230	3.152.221	02 min
	Ray	472	14.428	2.791.853	1h 26 min
	SOAPdenovo2	150.116	567	7.383.026	06 min
	SPAdes	1.971	13.028	2.786.068	16 min
	Velvet	60.300	114	3.349.641	05 min

Fonte – Próprio autor.

Tabela 5 – *k-mer* 33

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	14.797	2.370	4.989.335	11 min
	Minia	55.681	3.196	6.452.337	02 min
	Ray	651	11.071	4.509.119	1h 21 min
	SOAPdenovo2	-	-	-	-
	SPAdes	3.287	32.880	4.553.942	12 min
	Velvet	231.392	128	23.183.636	6 min
SRR1748018	ABySS	4.081	5.747	4.428.869	02 min
	Minia	5.262	8.344	4.721.741	01 min
	Ray	2.924	11.697	4.679.370	38 min
	SOAPdenovo2	39.452	1.398	7.440.729	<01 min
	SPAdes	2.986	13.305	4.740.141	12 min
	Velvet	56.581	156	7.088.581	02 min
SRR3312980	ABySS	1.372	2.416	457.692	02 min
	Minia	3.635	3.056	927.085	01 min
	Ray	3.457	2.988	1.009.127	43 min
	SOAPdenovo2	22.480	550	1.632.366	<01 min
	SPAdes	386	2.426	566.076	11 min
	Velvet	33.283	127	3.228.782	03 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	20.306	2.023	5.720.189	07 min
	Minia	7.272	4.282	5.247.385	01 min
	Ray	2.141	4.787	5.161.169	1h 34 min
	SOAPdenovo2	52.002	934	7.424.395	05 min
	SPAdes	2.333	7.566	5.195.722	31 min
	Velvet	72.945	136	6.976.396	06 min
SRR1769012	ABySS	35.892	1.727	7.444.015	16 min
	Minia	33.293	3.127	7.289.229	06 min
	Ray	631	19.685	6.628.898	5h 42 min
	SOAPdenovo2	169.891	628	14.062.339	27 min
	SPAdes	1.000	45.433	6.012.092	1h
	Velvet	124.505	118	10.820.905	58h 5 min
ERR493460	ABySS	15.437	6.841	3.398.195	05 min
	Minia	7.530	11.138	3.038.210	02 min
	Ray	310	19.176	2.846.124	1h 22 min
	SOAPdenovo2	83.258	666	6.733.223	07 min
	SPAdes	1.028	34.822	2.787.674	16 min
	Velvet	41.609	115	3.613.698	05 min

Fonte – Próprio autor.

Tabela 6 – *k-mer* 55

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	18.754	1.663	5.680.024	18 min
	Minia	10.301	2.997	4.966.232	02 min
	Ray	2.015	2.477	3.699.784	59 min
	SOAPdenovo2	-	-	-	-
	SPAdes	743	71.521	4.477.192	14 min
	Velvet	70.577	159	10.982.932	45 min
SRR1748018	ABySS	2.557	5.492	4.417.496	02 min
	Minia	3.507	9.019	4.680.110	01 min
	Ray	1.494	4.358	4.324.298	37 min
	SOAPdenovo2	10.277	702	3.675.207	<01 min
	SPAdes	1.576	20.228	4.708.531	12 min
	Velvet	31.864	189	6.039.776	01 min
SRR3312980	ABySS	1.171	2.433	482.071	02 min
	Minia	2.229	6.113	725.849	01 min
	Ray	175	3.299	398.382	44 min
	SOAPdenovo2	5.593	779	797.749	<01 min
	SPAdes	166	5.448	468.734	11 min
	Velvet	24.470	148	3.616.688	03 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	17.564	1.908	6.011.410	07 min
	Minia	5.208	3.760	4.958.218	01 min
	Ray	2.340	2.777	4.751.242	1h 34 min
	SOAPdenovo2	22.997	1.216	6.467.755	02 min
	SPAdes	2.409	7.575	5.241.356	30 min
	Velvet	42.057	150	6.492.616	02 min
SRR1769012	ABySS	37.961	1.584	8.590.421	19 min
	Minia	31.754	3.579	8.121.649	05 min
	Ray	930	9.843	7.072.357	5h 42 min
	SOAPdenovo2	65.445	820	10.643.326	14 min
	SPAdes	600	68.513	6.022.656	1h
	Velvet	143.562	141	20.272.687	50h 5min
ERR493460	ABySS	10.548	9.320	3.494.742	06 min
	Minia	3.583	16.228	2.988.235	02 min
	Ray	964	4.013	3.085.620	1h 13 min
	SOAPdenovo2	25.618	1.276	4.568.703	02 min
	SPAdes	601	44.496	2.796.658	16 min
	Velvet	27.165	142	3.816.552	04 min

Fonte – Próprio autor.

Tabela 7 – *k-mer* 77

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	10.213	1.858	5.207.023	31 min
	Minia	6.329	1.838	4.331.364	01 min
	Ray	976	1.081	916.489	43 min
	SOAPdenovo2	-	-	-	-
	SPAdes	315	122.212	4.453.758	34 min
	Velvet	31.138	205	6.745.288	45 min
SRR1748018	ABySS	2.133	5.185	4.395.015	02 min
	Minia	3.417	4.742	4.597.050	01 min
	Ray	2.824	1.569	3.631.007	43 min
	SOAPdenovo2	153	5.231	48.531	<01 min
	SPAdes	983	37.955	4.662.428	12 min
	Velvet	21.416	243	5.391.582	01 min
SRR3312980	ABySS	1.011	3.459	499.120	03 min
	Minia	1.380	10.660	606.475	01 min
	Ray	150	3.105	398.875	44 min
	SOAPdenovo2	353	5.131	426.631	<01 min
	SPAdes	63	12.727	424.668	11 min
	Velvet	17.525	192	3.449.942	01 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	16.463	1.824	6.330.794	08 min
	Minia	5.158	3.201	4.713.779	02 min
	Ray	2.930	1.624	4.111.987	1h 24 min
	SOAPdenovo2	11.137	979	4.874.703	<01 min
	SPAdes	2.614	7.652	5.295.174	31 min
	Velvet	25.626	202	5.623.224	03 min
SRR1769012	ABySS	-	-	-	-
	Minia	32.981	3.565	9.144.001	06 min
	Ray	1.722	4.972	6.699.257	5h 35 min
	SOAPdenovo2	12.432	1.875	7.091.089	05 min
	SPAdes	389	84.357	6.023.637	1h 3 min
	Velvet	-	-	-	-
ERR493460	ABySS	7.761	9.696	3.516.915	06 min
	Minia	1.979	15.669	2.927.115	02 min
	Ray	1.879	1.843	2.952.375	1h 13 min
	SOAPdenovo2	6.470	2.796	3.284.279	02 min
	SPAdes	376	86.440	2.797.535	16 min
	Velvet	18.488	184	3.453.274	04 min

Fonte – Próprio autor.

Tabela 8 – *k-mer* 99

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	14.666	1.280	6.119.696	10 min
	Minia	6.484	945	3.528.186	01 min
	Ray	130	741	89.272	40 min
	SOAPdenovo2	-	-	-	-
	SPAdes	121	61.870	4.386.442	31 min
	Velvet	11.811	433	4.464.778	02 min
SRR1748018	ABySS	1.954	4.581	4.346.063	02 min
	Minia	3.645	2.198	4.299.856	01 min
	Ray	2.249	977	1.960.391	43 min
	SOAPdenovo2	-	-	-	-
	SPAdes	660	63.800	4.624.317	12 min
	Velvet	13.122	392	4.731.336	01 min
SRR3312980	ABySS	763	3.841	494.818	03 min
	Minia	710	10.912	510.599	01 min
	Ray	160	2.885	414.683	45 min
	SOAPdenovo2	-	-	-	-
	SPAdes	35	18.173	410.637	11 min
	Velvet	9.719	225	2.340.762	01 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	15.968	1.789	6.675.269	07 min
	Minia	3.645	2.198	4.299.856	02 min
	Ray	2.455	1.321	2.890.206	1h 17 min
	SOAPdenovo2	-	-	-	-
	SPAdes	2.640	7.680	5.347.324	31 min
	Velvet	15.296	335	4.910.440	03 min
SRR1769012	ABySS	-	-	-	-
	Minia	30.395	3.748	9.764.229	07 min
	Ray	2.578	2.816	6.155.629	5h 7 min
	SOAPdenovo2	-	-	-	-
	SPAdes	304	119.461	6.031.314	1h 11 min
	Velvet	-	-	-	-
ERR493460	ABySS	6.476	8.245	3.572.580	05 min
	Minia	1.261	13.012	2.849.285	01 min
	Ray	1.781	1.410	2.257.827	58 min
	SOAPdenovo2	-	-	-	-
	SPAdes	383	125.850	2.812.170	16 min
	Velvet	11.850	230	3.025.707	02 min

Fonte – Próprio autor.

Tabela 9 – *k-mer* 127

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
SRR788730	ABySS	9.293	812	4.722.744	11 min
	Minia	4.631	617	1.745.486	01 min
	Ray	-	-	-	-
	SOAPdenovo2	-	-	-	-
	SPAdes	228	28.010	4.361.402	31 min
	Velvet	3.906	633	2.426.199	02 min
SRR1748018	ABySS	3.209	2.508	4.551.206	02 min
	Minia	5.798	984	3.789.233	01 min
	Ray	463	737	342.430	18 min
	SOAPdenovo2	-	-	-	-
	SPAdes	350	224.364	4.570.017	12 min
	Velvet	4.815	1.090	4.182.506	01 min
SRR3312980	ABySS	607	5.204	494.100	03 min
	Minia	536	19.293	512.284	01 min
	Ray	170	2.679	419.433	51 min
	SOAPdenovo2	-	-	-	-
	SPAdes	13	107.421	405.969	11 min
	Velvet	3.280	254	943.961	01 min

(continuação)

Bibliotecas SRA	Montadores	Número de *contigs*	N50	*Total Length*	Tempo de execução
ERR732677	ABySS	13.412	1.741	6.638.272	07 min
	Minia	4.525	1.776	4.042.348	02 min
	Ray	1.108	1.130	1.178.204	1h 24 min
	SOAPdenovo2	-	-	-	-
	SPAdes	1.430	9.608	5.276.962	31 min
	Velvet	6.411	595	3.452.199	02 min
SRR1769012	ABySS	-	-	-	-
	Minia	24.842	4.029	9.967.532	05 min
	Ray	2.713	2.056	5.080.572	4h 42 min
	SOAPdenovo2	-	-	-	-
	SPAdes	125	242.633	6.082.111	1h 18 min
	Velvet	-	-	-	-
ERR493460	ABySS	5.492	6.954	3.601.102	05 min
	Minia	1.707	3.364	2.699.298	01 min
	Ray	734	1.400	936.453	47 min
	SOAPdenovo2	-	-	-	-
	SPAdes	143	204.059	2.796.238	16 min
	Velvet	4.531	649	2.465.771	01 min

Fonte – Próprio autor.

Anexos

ANEXO A – Instalação e configuração dos programas utilizados no estudo

Pretendemos neste anexo descrever os comandos utilizados para a instalação e configuração dos programas presentes neste estudo, bem como os parâmetros aplicados em suas execuções. Entretanto, não iremos descrever detalhadamente o funcionamento desses códigos, uma vez que estes já foram descritos nos manuais dos respectivos *softwares*.

A.1 ABySS

A.1.1 Instalação e configuração

Para ser executado corretamente, o ABySS requer a instalação das seguintes bibliotecas: *Boost*, *Open MPI* e *sparsehash*.

Instalação do *Boost*:

```
sudo apt-get install libboost-all-dev
```

Ao instalar a dependência *Boost*, o gerenciador de pacotes do Linux instala automaticamente outra biblioteca, a *Open MPI*. Entretanto, a versão do *Open MPI* disponibilizada pelo gerenciador é antiga, o que, porém, não impediu a execução do programa de montagem. Contudo, a fim de promover o melhor cenário para a ferramenta, instalamos a versão mais atual da biblioteca por meio dos seguintes comandos:

```
wget https://www.open-mpi.org/software/ompi/v4.0/downloads/openmpi-4.0.4.tar.gz
tar -xvf openmpi-4.0.4.tar.gz
cd openmpi-4.0.4
./configure –prefix="/home/$USER/.openmpi"
make
sudo make install
export PATH="PATH : /home/USER/.openmpi/bin"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/$USER/.openmpi/lib
```

Instalação do *sparsehash*:

```
sudo apt-get install sparsehash
```

Após a instalação das dependências, basta instalar o ABySS com o seguinte comando:

```
sudo apt-get install abyss
```

A.1.2 Parâmetros de execução

```
abyss-pe name=<nome_dos_arquivos_de_saída> k=<valor_do_kmer> se=<diretório_da_leitura> v=-v
```

A.2 Minia

A.2.1 Instalação e configuração

Do mesmo modo que o ABySS, o Minia também requer a instalação de pré-requisitos, mais especificamente do compilador *Cmake*. Para instalá-lo, utilizamos os seguintes comandos:

```
wget https://github.com/Kitware/CMake/releases/download/v3.15.2/cmake-3.15.2.tar.gz
tar -zxvf cmake-3.15.2.tar.gz
cd cmake-3.15.2
./bootstrap
make
sudo make install
```

Com o compilador pronto para uso, a instalação do Minia se deu da seguinte maneira:

```
git clone –recursive https://github.com/GATB/minia.git
cd minia
sh INSTALL
cp /minia/build/bin/minia /usr/local/bin
```

A.2.2 Parâmetros de execução

```
minia -in <diretório_da_leitura> -kmer-size <valor_do_kmer> -out <diretório_de_saída>
```

A.3 MIRA

A.3.1 Instalação e configuração

O MIRA também possui dependências que precisam ser atendidas antes da sua instalação, sendo elas: *GNU Compiler Collection* (GCC), biblioteca *Booster* e o pacote *"libexpat1-dev"*.

Instalação do GNU *Compiler Collection* (GCC):

```
sudo apt-get install gcc make flex
```

A instalação da biblioteca *Booster* já foi descrita anteriormente. Assim, resta somente o pacote *"libexpat1-dev"*, que foi instalado por meio do seguinte comando:

```
sudo apt-get install libexpat1-dev
```

Após atender todos os pré-requisitos, basta instalar o MIRA via terminal:

```
git clone https://github.com/bachev/mira.git
cd mira
./bootstrap.sh
./configure
make
make install
```

A.3.2 Parâmetros de execução

O MIRA trabalha com arquivos de configuração como metodologia principal de execução. Esses arquivos são chamados de *"manifest.conf"* e possuem por objetivo informar o tipo de montagem a ser realizada e os dados a serem carregados. A sintaxe utilizada no arquivo de configuração foi:

```
project = <nome_dos_arquivos_de_saída>
job = denovo,genome,accurate
parameters = COMMON_SETTINGS
readgroup = IonTorrent
data = <diretório_das_leituras>
technology = iontor
segment_naming = sra
```

O arquivo de manifesto é dividido em duas partes, sendo a primeira delas destinada as configurações gerais do processo, como o nome do projeto, os parâmetros utilizados e o tipo de montagem – definido em *"job"* como *denovo, genome e acurate*. Na segunda parte é definido as características das leituras, iniciando pelo chamado grupo de leitura *(readgroup)* – escolhido de modo arbitrário pelo usuário -, diretório dos arquivos a serem montados *(data)* e a sua tecnologia de sequenciamento – indicado por *"technology"* como *Ion Torrent*. Por fim, é determinado em *"segment_naming"* o esquema de nomes que as leituras estão seguindo para indicar o modelo de DNA a qual pertencem (CHEVREUX et

al., 2004).

O MIRA é um *software* altamente sensível a qualidade e configuração dos dados de entrada. Por padrão, caso ele identifique qualquer anormalidade nos arquivos o processo de montagem é abortado. Assim sendo, os parâmetros foram utilizados de modo a se adaptarem as particularidades dos genomas e possibilitar a sua execução.

Durante a montagem da biblioteca SRR3312980, por exemplo, o programa identificou uma alta cobertura média no arquivo de entrada, que aparentemente excedia o limite do *software*. Em vista disso, utilizamos como parâmetro o comando *"NW:cac=no"* para indicar ao programa a realização da montagem mesmo com a limitação presente no arquivo. No restante das execuções, o MIRA identificou uma anormalidade na nomeação dos modelos de leitura, fato esse que também o levou a abortar o processo. No entanto, sabendo que estes são fatores decorrentes da etapa de sequenciamento, usamos o parâmetro *"-NW:ctp=no"* para dar prosseguimento à montagem.

Por fim, o último parâmetro utilizado foi o *"-AS:kms="*, sendo responsável por indicar o valor de *k-mer* a ser utilizado no estágio de pré-montagem. Como dito no capítulo 3, o MIRA adota a frequência de *k-mer* como um critério para identificar a repetitividade de uma determinada sequência de DNA (CHEVREUX et al., 2004). Por padrão, o próprio montador escolhe o tamanho de K, como foi observado na maioria das execuções. Contudo, para a realização de algumas dessas montagens foi necessário atribuir o valor manualmente, possivelmente devido a alguma anormalidade nos arquivos de entrada. Assim, a utilização desse comando variou de acordo com a biblioteca montada.

A.4 Ray

A.4.1 Instalação e configuração

O Ray é um *software* que, como os outros, também necessita da instalação de pré-requisitos para funcionar corretamente, especialmente de um compilador C++ - como o *Cmake* - e de uma biblioteca MPI - como o *Open MPI*. A instalação desses dois programas já foi descrita nas seções anteriores.

Por padrão, este montador possui um tamanho limitado para o parâmetro *k-mer*, com 31 sendo o valor máximo. Logo, o Ray foi compilado de modo a aceitar tamanhos maiores para esse parâmetro.

Após todas as dependências atendidas, basta fazer o *download* do montador e, no diretório em que o arquivo foi salvo, inserir o seguinte código no terminal:

```
tar xjf Ray-2.3.1.tar.bz2
cd Ray-2.3.1
make PREFIX=Ray-Large-k-mers MAXKMERLENGTH=128
make install
sudo cp /home/.../Downloads/Ray-2.3.1/Ray-Large-k-mers/Ray /usr/bin/
```

A.4.2 Parâmetros de execução

```
Ray -k <valor_do_kmer> -s <diretório_da_leitura> -o <diretório_de_saída>
```

A.5 SOAPdenovo2

A.5.1 Instalação e configuração

A instalação do*SOAPdenovo2* depende unicamente do GNU *Compiler Collection* (GCC), que normalmente vem instalado por padrão nos sistemas Linux Ubuntu. Caso contrário, basta instalá-lo seguindo as orientações feitas nas seções anteriores. Isto feito, resta somente instalar o montador:

```
git clone https://github.com/aquaskyline/SOAPdenovo2.git
cd SOAPdenovo2
make
sudo cp SOAPdenovo127-mer /usr/bin/
```

A.5.2 Parâmetros de execução

No que tange ao fluxo de trabalho, o SOAPdenovo2 pode ser considerado um programa híbrido, já que seu funcionamento depende tanto de recursos da linha de comando quanto do arquivo de configuração. Este arquivo divide os dados de entrada em bibliotecas (LIB) para melhor organizá-los. A sintaxe utilizada foi:

```
[LIB]
asm_flags=1
q=<diretório_da_leitura>
```

O parâmetro *"asm_flags"* indica o modo como as leituras serão usadas. No nosso caso, utilizamos o valor 1 para indicar ao programa que só desejamos obter *contigs*. O parâmetro "q" indica o diretório em que as leituras estão armazenadas e, simultaneamente, o formato no qual os dados foram obtidos *(fastq)*.

Quanto ao recurso da linha de comando, a sintaxe utilizada foi:

```
SOAPdenovo-127mer all -s <diretório_do_arquivo_de_configuração> -K <valor_do_kmer> -R -o <nome_dos_arquivos_de_saída> 1>ass.log 2>ass.err
```

A.6 SPAdes

A.6.1 Instalação e configuração

Para instalar o SPAdes é necessário, primeiramente, instalar o Python no sistema operacional. Para isso, basta somente seguir os seguintes comandos:

```
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar xvzf Python-3.5.2.tgz
cd Python-3.5.2
./configure
make
make test
sudo make install
```

Isto feito, o próximo passo é instalar o SPAdes:

```
wget http://cab.spbu.ru/files/release3.14.1/SPAdes-3.14.1-Linux.tar.gz
tar -xzf SPAdes-3.14.1-Linux.tar.gz
cd SPAdes-3.14.1-Linux/bin/
sudo cp -a bin/* /usr/local/bin
sudo cp -a share/spades/ /usr/local/share/
```

A.6.2 Parâmetros de execução

```
spades.py -k <valor_do_kmer> –iontorrent -s <diretório_da_leitura> -o <diretório_de_saída>
```

A.7 Velvet

A.7.1 Instalação e configuração

Para ser executado corretamente, o montador precisa da biblioteca *zlib* instalada na máquina:

```
sudo apt-get install zlib1g-dev
```

Ademais, do mesmo modo que a ferramenta Ray, o Velvet também possui um valor máximo para o parâmetro *k-mer*. Assim sendo, o programa precisou ser compilado de modo a aceitar valores maiores nesse parâmetro.

```
wget http://www.ebi.ac.uk/ zerbino/velvet/velvet_1.2.10.tgz
tar xvzf velvet_1.2.10.tgz
cd velvet_1.2.10
make
make 'MAXKMERLENGTH=127'
make 'CATEGORIES=57'
sudo cp velvetg velveth /usr/local/bin/
```

A mesma configuração pode ser feita diretamente no arquivo de compilação do programa.

A.7.2 Parâmetros de execução

A execução do Velvet é composta por dois módulos: *velveth* e *velvetg*. O primeiro é responsável por construir e preparar as leituras para a etapa da montagem, indicando, por exemplo, o seu tipo e formato. A sintaxe utilizada no *velveth* foi:

```
velveth <diretório_de_saída> <valor_do_kmer> fastq short <diretório_da_leitura>
```

Após a conclusão dessa etapa, o segundo módulo *(velvetg)* utiliza os arquivos gerados pelo processo anterior para executar a montagem. Portanto, para a sua execução, é necessário somente indicar o diretório de saída gerado pelo velveth:

```
velvetg <diretório_de_saída>
```

A.8 QUAST

A.8.1 Instalação e configuração

Para a correta execução do QUAST, algumas dependências precisam ser atendidas, como: Python, GNU *Compiler Collection* (GCC), biblioteca *zlib* entre outros. A maioria desses requisitos já vem pré-instalados no Linux Ubuntu ou já tiveram o seu processo de instalação descrito nesse apêndice.

Além desses programas, o QUAST também exige a instalação de uma biblioteca chamada *"Matplotlib"*, que geralmente não vem pré-instalada no Ubuntu. Assim, para

instalá-la, utilizamos o seguinte comando:

```
sudo pip install matplotlib
```

Em seguida, basta instalar o QUAST:

```
wget https://downloads.sourceforge.net/project/quast/quast-5.0.2.tar.gz
tar -xzf quast-5.0.2.tar.gz
cd quast-5.0.2
./setup.py install_full sudo cp quast.py /usr/local/bin/
```

A.8.2 Parâmetros de execução

```
quast.py -o <diretório_de_saída> <diretório_do_arquivo_de_montagem>
```

Por padrão, o QUAST somente avalia *contigs* maiores que 500 pares de base, o que julgamos aceitável para a maioria dos montadores. No entanto, levando em consideração que a arquitetura do Velvet é voltada para leituras extremamente curtas, tivemos que reduzir o valor padrão do parâmetro *"min-contigs"* para 100 ao avaliar a sua montagem, como descrito abaixo:

```
quast.py -o <diretório_de_saída> –min-contig 100 <diretório_do_arquivo_de_montagem>
```