Pontifícia Universidade Católica
do Rio de Janeiro

Alejandro Mustelier Menés

Avaliação da qualidade da montagem de fragmentos de sequências biológicas

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Sérgio Lifschitz

Rio de Janeiro
Outubro de 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Alejandro Mustelier Menés

Avaliação da qualidade da montagem de fragmentos de sequências biológicas

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

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

Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Marcos Paulo Catanho de Souza
Laboratório de Genômica Funcional e
Bioinformática – Fundação Oswaldo Cruz

Kary Ann del Carmen Ocaña Gautherot
Laboratório de Bioinformática – LNCC

Prof. Marcio da Silveira Carvalho
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 6 de outubro de 2017

Alejandro Mustelier Menés

Graduou-se em Ciências da Computação na Universidade de Havana (UH), Havana –Cuba em 2013. Começou o Mestrado em Informática na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) em 2014.

Ficha Catalográfica

Menés, Alejandro Mustelier

Avaliação da qualidade da montagem de fragmentos de sequências biológicas / Alejandro Mustelier Menés; orientador: Sérgio Lifschitz. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2017.

v., 61 f.: il. ; 29,7 cm

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

Inclui referências bibliográficas.

1. Informática – Teses. 2. Bioinformática. 3. Montagem de Fragmentos. 4. Sequências Biológicas. I. Lifschitz, Sérgio. II Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador por sua paciência e ajuda, essenciais para lidar com os momentos diferentes pelos quais passei durante o curso.

Agradeço aos meus pais e amigos, pelo incentivo, carinho e pelos maravilhosos momentos de descontração. O apoio de todos foi fundamental para o desenvolvimento e conclusão dessa dissertação. Sem o apoio deles desde cedo, não teria chegado até aqui.

A minha esposa por toda sua ajuda.

Aos meus colegas da PUC-Rio.

Aos professores que participaram da Comissão examinadora

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Resumo

Menés, Alejandro Mustelier; Lifschitz, Sérgio. **Avaliação da qualidade da montagem de fragmentos de sequências biológicas**. Rio de Janeiro, 2017. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nos últimos anos surgiram novas tecnologias de sequenciamento de DNA conhecidas como NGS - *Next-Generation Sequencing*. Estas são responsáveis por tornar o processo de sequenciamento mais rápido e menos custoso, mas também trazem como resultado fragmentos de DNA muito pequenos, conhecidos como *reads*. A montagem do genoma a partir destes fragmentos é considerada um problema complexo devido à sua natureza combinatória e ao grande volume de *reads* produzidos. De maneira geral, os biólogos e bioinformatas escolhem o programa montador de sequências sem levar em consideração informações da eficiência computacional ou da qualidade biológica do resultado. Esta pesquisa tem como objetivo auxiliar aos usuários biólogos a avaliar a qualidade dos resultados da montagem. Primeiramente, foi projetada e desenvolvida uma metodologia para obter informações dos genes presentes na montagem, listando os genes que podem ser identificados, aqueles que têm o tamanho correto e a sequência de pares de bases correta. Em segundo lugar, foram realizados testes experimentais exaustivos envolvendo cinco dos principais montadores de genoma conhecidos na literatura os quais são baseados no uso de grafos de *Bruijn* e oito genomas de bactérias. Foram feitas comparações estatísticas do resultado usando as ferramentas *QUAST* e REAPR. Também foram obtidas informações qualitativas dos genes usando o algoritmo proposto e algumas métricas de eficiência. Em função dos resultados coletados, é feita uma análise comparativa que permite aos usuários conhecer melhor o comportamento das ferramentas consideradas nos testes. Por fim, foi desenvolvida uma ferramenta que recebe diferentes resultados de montagens de um mesmo genoma e produz um relatório qualitativo e quantitativo para o usuário interpretar os resultados de maneira integrada.

Palavras-chave

Bioinformática; montagem de fragmentos; sequências biológicas.

Abstract

Menés, Alejandro Mustelier; Lifschitz, Sérgio (Advisor). **Quality evaluation for fragments assembly of biological sequences**. Rio de Janeiro, 2017. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

New DNA sequencing technologies, known as NGS - Next-Generation Sequencing, are responsible for making the sequencing process more efficient. However, they generate a result with very small DNA fragments, known as *reads*. We consider the genome assembly from these fragments a complex problem due to its combinatorial nature and the large volume of *reads* produced. In general, biologists and bioinformatics experts choose the sequence assembler program with no regard to the computational efficiency or even the quality of the biological result information. This research aims to assist users in the interpretation of assembly results, including effectiveness and efficiency. In addition, this may sometimes increase the quality of the results obtained. Firstly, we propose an algorithm to obtain information about the genes present in the result assembly. We enumerate the identified genes, those that have the correct size and the correct base pair sequence. Next, exhaustive experimental tests involving five of the main genome assemblers in the literature which are based on the use of graphs of *Bruijn* and eight bacterial genomes data set were ran. We have performed statistical comparisons of results using *QUAST* and REAPR tools. We have also obtained qualitative information for the genes using the proposed algorithm and some computational efficiency metrics. Based on the collected results, we present a comparative analysis that allows users to understand further the behavior of the tools considered in the tests. Finally, we propose a tool that receives different assemblies of the same genome and produces a qualitative and quantitative report for the user, enabling the interpretation of the results in an integrated way.

Keywords

Bioinformatics; fragments assembly; biological sequences.

Sumário

Lista de figuras

Lista de Tabelas

1
Introdução

1.1.
Motivação

Nos últimos anos têm surgido uma geração de tecnologias de sequenciamento de DNA conhecida como *Next-Generation Sequencing* (NGS) as quais foram desenvolvidas para reduzir o custo e o tempo do processo de sequenciamento. Elas geram uma grande massa de dados, a um custo relativamente baixo e com alta vazão. No entanto, as leituras curtas, por elas produzidas, dificultam sobre maneira o processo de montagem de genomas. Esse conjunto de tecnologias tem acelerado as pesquisas nas áreas biológicas, possibilitando a análise detalhada de organismos (SHENDURE; JI, 2008).

As tecnologias de sequenciamento até então eram, em sua maioria, baseadas no método de *Sanger* (SANGER, 1977), tendo como principais limitações o custo, tempo, velocidade e a escalabilidade. Tais barreiras foram superadas pelas *NGS* que utilizam uma abordagem diferente para o sequenciamento, o que desencadeou várias vantagens. Nos *NGS*, o processo de sequenciamento é muito mais rápido e mais barato, mas também traz *reads* (fragmentos) muito menores do que aqueles obtidos na tecnologia *Sanger*. Além do maior número de fragmentos, a inferência dos erros de sequenciamento faz com que o processo da montagem seja mais complicado (NAGARAJAN; POP, 2009).

Além disso existe uma plataforma *NGS* chamada *PacBio*, que produz reads com 10.000 pb em média, que é muito utilizada, apesar de sua alta taxa de erro (~14%); melhores montagens são obtidas quando são feitas abordagens híbridas de sequenciamento, usando *Illumina* e *PacBio* (RHOADS et al., [s.d.]).

Para resolver essa situação, os montadores de fragmentos de DNA atuais executam uma série de pré-processamentos e pós-processamentos nos dados a fim de eliminar, ou pelo menos diminuir, os erros e alcançar um resultado com um nível de confiança maior.

A montagem de *reads* é o ponto de partida dos trabalhos de pós-sequenciamento. A resolução dessa tarefa é um dos problemas fundamentais, pois é usada para conhecer a sequência de DNA de um indivíduo, determinar mutações entre indivíduos e achar novos métodos de diagnósticos para doenças, criação de vacinas, entre outros (SHENDURE; BIOTECHNOLOGY; 2012, [s.d.]).

A montagem de *reads* é considerada um problema complexo devido à sua natureza combinatória e ao grande volume de dados produzido pelas tecnologias de sequenciamento (LUQUE; ALBA, 2005). Os sequenciadores de DNA não conseguem trabalhar com a molécula completa de uma vez, fazendo com que um grande número de cópias da mesma molécula de DNA seja aleatoriamente dividido em fragmentos e que estes fragmentos, de diferentes cópias, se sobreponham parcialmente (LI; HOMER, 2010). A montagem computacional do DNA é atualmente o único meio de obter a sequência completa de uma molécula de DNA. Duas abordagens se destacam neste cenário: o grafo de sobreposição e o grafo de Bruijn (CHERUKURI; JANGA, 2016; JÜNEMANN et al., 2014)

Atualmente são os biólogos os que decidem qual montador usar e eles também avaliam a qualidade da resposta dada por este. Os biólogos não têm informação da eficiência computacional de um determinado montador para um tipo específico de genoma. Embora existam métricas e informações estatísticas (GUREVICH et al., 2013; HUNT; KIKUCHI; SANDERS, 2013) para avaliar a qualidade de uma montagem, elas não são integralmente usadas. Como consequência, o biólogo pode ficar com montagens de baixa qualidade ou gastar muito tempo para obter uma montagem de boa qualidade.

1.2. Objetivos e Contribuições

O objetivo desta dissertação é comparar os resultados de montagens usando critérios estatísticos e biológicos para tentar ajudar ao biólogo a determinar a qualidade.

Como objetivos específicos temos:

1. Selecionar métricas para analisar o resultado de uma montagem.
2. Obter e computar estatísticas que contribuam ao entendimento e análise do resultado da montagem.

3. Avaliar como o valor *k* influencia no resultado da montagem.
4. Desenvolver um software que calcule e mostre automaticamente as métricas que foram selecionadas.

Para alcançar o nosso objetivo foram realizados numerosos testes permitindo comparar a eficiência e o tempo de execução, a extração de estatísticas sobre a montagem e estudo dos genes presentes nos resultados e a avaliação da qualidade da montagem.

As estatísticas para avaliar a montagem foram obtidas usando as ferramentas *QUAST* e REAPR. Mesmo que com essas ferramentas consigamos um conjunto de avaliações importantes, na interação com usuários biólogos detectamos que o interesse de obter informações dos genes presentes na montagem; como quantos e quais genes, do genoma de referência, podem ser identificados numa montagem; se os genes identificados têm o tamanho correto e/ou têm a sequência de bases correta. Assim, a metodologia para realizar essas análises é proposta e desenvolvida neste trabalho.

Também foi realizada uma série de testes, envolvendo cinco dos principais montadores de genoma conhecidos na literatura e oito genomas de bactérias, que vão ser a base para sugerir montadores e parâmetros biológicos “padrões” para um determinado tipo de dado.

Por último é proposta uma ferramenta que acopla os diferentes montadores assim como a avaliação do resultado em um modulo só. A ferramenta também combina as diferentes análises para criar novos relatórios, o que ajuda na escolha da melhor montagem de forma eficiente.

1.3. Estrutura da Dissertação

Esta dissertação está estruturada da seguinte forma. O capítulo 2 fornece uma visão geral dos principais conceitos relacionados ao tema da dissertação. O capítulo 3 resume os trabalhos relacionados. O capítulo 4 apresenta o funcionamento dos montadores selecionados e as análises propostas. O capítulo 5 traz os resultados dos testes feitos. O capítulo 6 detalha a implementação da ferramenta. Finalmente, o capítulo 7 apresenta as conclusões e propõe os trabalhos futuros.

2
Contexto

Este capítulo fornece uma visão geral dos principais conceitos relacionados a esta dissertação. Como os sequenciadores de DNA não conseguem trabalhar com a molécula completa de uma vez, ela é quebrada por meio de enzimas, para o tamanho desejado de read por exemplo 100pb um tamanho que o sequenciador consegue sequenciar. O processo de sequenciamento trabalha com diversos fragmentos que precisam ser montados em um próximo passo. As Seções 2.1 e 2.2 são, respectivamente uma síntese dessas duas etapas. Nessas seções apresentamos a importância de alguns parâmetros e as diferenças entre os montadores. Finalmente, a Seção 2.3 esclarece quando e quais erros aparecem na etapa de montagem de sequências.

2.1.
Processo de Sequenciamento

O sequenciamento consiste na leitura e descodificação dos nucleotídeos de DNA que compõem o genoma de um organismo vivo. As tecnologias de sequenciamento possuem limitação no tamanho dos fragmentos que processam, não conseguindo, ainda, sequenciar genomas complexos de forma única (MILLER; KOREN; SUTTON, 2010). Devido ao fato das tecnologias atuais de sequenciamento não serem capazes de sequenciar um genoma inteiro de uma vez só, é necessário que diversas cópias sejam produzidas e fragmentadas, para que tais fragmentos tenham tamanhos adequados para serem sequenciados. Uma vez sequenciados os fragmentos, obtém-se um conjunto de *reads*. Estes precisam, por fim, serem montados para que a sequência da molécula de DNA inicial seja descoberta. Com a presença de sequências de tamanhos reduzidos as chances de acontecerem situações ambíguas na ordem em que as sequências são remontadas é muito maior (SHENDURE; JI, 2008).

As *NGSs* permitem a geração de *reads* em grande velocidade resultando em uma grande quantidade de *reads* os quais geralmente tem comprimento pequeno (*short reads*) (DAVEY et al., 2011). O sucesso das tecnologias de sequenciamento não depende apenas da rapidez do processo de produção dos *reads*, senão da solução ao problema de montagem de fragmentos em genomas completos. A dificuldade em analisar as repetições faz com que a montagem de genomas longos utilizando *reads* curtos seja um problema computacionalmente difícil (CHAISSON; PEVZNER; TANG, 2004).

2.2. Montagem de genomas

O resultado obtido após o sequenciamento do genoma pelo sequenciador que corresponde a uma pequena sequência de poucos pares de base é conhecido como *reads*. Os *contigs* são um conjunto de segmentos contíguos formados a partir da sobreposição dos *reads*. Por sua parte os *scaffolds* são um conjunto de dois ou mais *contigs* de forma ordenada e orientada de acordo com a estrutura do genoma original, pode ter áreas faltantes conhecidas como gaps.

O processo de montagem consiste na sobreposição dos *reads* a fim de formar *contigs*, para reconstruir uma sequência de DNA original de um organismo sem introduzir erros. O desafio principal da montagem do genoma é tratar as sequências repetitivas (SCHATZ; DELCHER; SALZBERG, 2010). A magnitude do desafio depende da tecnologia de sequenciamento porque a fração de *reads* repetitivos depende do comprimento dos próprios *reads*. No entanto, os genomas reais têm estruturas de repetição complicadas, fazendo com que algumas sequências sejam quase impossíveis de serem montadas corretamente (SCHATZ; DELCHER; SALZBERG, 2010).

Basicamente, há dois tipos de montagem de fragmentos de DNA (PASZKIEWICZ; STUDHOLME, 2010; POP, 2009; ZHANG et al., 2011). O primeiro é conhecido como *Mapping* e usa um genoma de referência. Este tipo consiste no re-sequenciamento que ocorre quando o pesquisador tem à disposição durante a montagem de uma espécie X_a o genoma previamente montado de uma outra espécie X_b e ambas são notavelmente similares. Sendo assim, parte do processo de montagem está na comparação de áreas semelhantes dos dois

genomas para alinhamento das sequências. Neste caso, as leituras do sequenciamento não são comparadas entre si, mas, sim alinhadas contra a referência, o resultado da sobreposição dos alinhamentos é usado para a sequência de consenso. A sequência gerada normalmente é muito semelhante a original, mas não idêntica.

O segundo tipo de montagem é conhecido como *de novo*. Nas montagens *de novo* apenas a informação obtida dos sequenciadores de DNA é conhecida, sem informação extra de outras espécies (BAKER, 2012; LI et al., 2010). A montagem é realizada identificando sobreposições entre os *reads*, e em seguida, são gerados *contigs* usando sequências consenso. Posteriormente, vários *contigs* podem ser unificados em *scaffolds*, usando informações de união para gerar sequências ainda maiores.

A montagem de tipo *de novo* é computacionalmente muito mais complexa, requerendo muito mais recursos (BAKER, 2012). Sendo considerada um problema NP-Difícil (NAGARAJAN; POP, 2009; POP, 2009). Para tratar as topologias problemáticas, os montadores de fragmentos *NGS* executam diversos pré e pós-processamentos nos *reads*, tornando o processo de montagem mais custoso.

Em relação ao processo de montagem *de novo*, existem duas abordagens principais (CONWAY; BROMAGE, 2011), sendo elas a estratégia de *Overlap-Layout-Consensus* (*OLC*) e os grafos de Bruijn. Neste trabalho nos focaremos nesta última abordagem.

O Grafo de Bruijn tem esse nome em homenagem ao matemático holandês Nicolaas Govert de Bruijn que, em 1946, interessou-se em resolver o problema de encontrar a menor supersequência circular que contenha todas as subsequências de tamanho *k* (*k-mers*) sobre um alfabeto β (COMPEAU; PEVZNER; TESLER, 2011).

Existem n^k k-*mers* em um alfabeto contendo *n* símbolos. Por exemplo dado um alfabeto $\beta=\{A, C, G, T\}$ e $k=3$, temos $4^3= 64$ triplas, ou seja, *64* possíveis *3-mers*.

Nicolaas Govert de Bruijn criou uma solução para o problema baseada em caminho Euleriano, onde é construído um grafo *B*, onde para cada *(k-1)-mer* é criado um vértice. Um vértice *x* se conecta a outro vértice *y* através de uma aresta dirigida, se existir uma sobreposição de tamanho *k-2* do sufixo do vértice *x* com o prefixo do vértice *y*. As arestas do grafo *de Bruijn* representam todos os *k-mers*.

Um caminho Euleriano no grafo representa uma supersequência que contém exatamente cada *k-mer* uma única vez.

A primeira aplicação dos grafos *de Bruijn* para a montagem do genoma foi proposta por Pevzner em 2001 (BAKER, 2012). Ao converter o conjunto de leituras em arestas do grafo *de Bruijn*, o problema da montagem torna-se equivalente a encontrar um caminho Euleriano no grafo. Entretanto, pode haver um número exponencial de caminhos Eulerianos distintos em um grafo, enquanto que apenas um pode ser considerado como a montagem correta do genoma. Para reduzir a complexidade do problema, heurísticas são geralmente aplicadas ao grafo construído.

Entre os montadores que podemos categorizar neste grupo, que utilizam a abordagem baseada no grafo de *Bruijn* podemos encontrar: Minia (CHIKHI; RIZK, 2012), Velvet (ZERBINO; BIRNEY, 2008), SOAPdenovo2 (LUO et al., 2012), SPAdes (BANKEVICH; NURK; ANTIPOV, 2012a) e MaSuRCA (ZIMIN et al., 2013). Os montadores anteriormente citados serão usados no nosso trabalho por ser dos mais representativos na literatura e ser utilizados em projetos de referências como *GAGE* (MAGOC et al., 2013; SALZBERG et al., 2012) e *Assemblathon* (EARL et al., 2011). A Seção 4.1 detalha as principais características destes montadores.

2.3. Limitações dos Montadores

As técnicas para montagem da sequência genômica a partir de *reads* de sequenciamento curtos tem diversas limitações que impedem a reconstrução correta e completa da sequência dos genomas. Estas limitações não são exclusivamente causadas pela ineficiência dos atuais métodos, mas são altamente influenciadas pelas características dos dados. Aproximadamente 30% das sequências de um genoma são repetidas. Os paradigmas para montagem de genoma abstraem os dados em grafos e, quando regiões de repetição ocorrem, o grafo gera caminhos ambíguos, impossibilitando a montagem correta (NAGARAJAN; POP, 2013). Quanto menor o tamanho do *read*, mais complexo será descobrir do caminho correto no grafo gerado e mais difícil será a resolução de regiões de repetição (KINGSFORD; SCHATZ; POP, 2010). Estas regiões de

repetição são o principal fator de limitação nas montagens genômicas. Os erros da montagem acontecem porque são descartadas, incorretamente, sequências como erros ou repetições, e outras se juntam em lugares ou orientações erradas (BAKER, 2012).

Neste capítulo foi feita a explicação do processo de sequenciamento e do processo de montagem de genomas, assim como algumas limitações que tem os montadores ao criar a montagem. Com o objetivo de avaliar a qualidade de uma montagem tem sido produzida diferentes pesquisas. O seguinte capítulo fornece uma visão geral das principais pesquisas.

3
Trabalhos Relacionados

Este capítulo fornece uma visão geral das diferentes pesquisas relacionadas à geração de montagens de genomas corretas. A Seção 3.1 descreve dois projetos de pesquisas para avaliar montadores. A Seção 3.2 descreve diferentes ferramentas que analisam as características específicas de uma montagem para estimar a qualidade do resultado. Os trabalhos foram selecionados pela sua relevância para a discussão e pelas contribuições para o desenvolvimento do trabalho atual.

3.1.
Avaliação de Montadores

O projeto *GAGE* (MAGOC et al., 2013; SALZBERG et al., 2012) consiste em uma avaliação dos algoritmos mais recentes de montagem de genoma. Eles organizam uma competição para produzir uma avaliação realista dos montadores de genomas e uma análise do comportamento dos montadores frente à rápida evolução dos *NGS*. Os resultados desta avaliação descrevem o desempenho relativo dos diferentes montadores, bem como outras diferenças significativas na dificuldade de montagem que parecem ser inerentes aos próprios genomas. Até o momento o projeto gerou três conclusões principais: primeiro, a qualidade dos dados, em vez do próprio montador, tem um efeito na qualidade de um genoma montado; segundo, o grau de contiguidade de uma montagem varia enormemente entre diferentes montadores e genomas diferentes; e terceiro, a correção de uma montagem também varia muito e não está bem correlacionada com estatísticas sobre a contiguidade.

O evento *Assemblathon* (EARL et al., 2011) surgiu de um conjunto de esforços colaborativos e periódicos para ajudar a melhorar os métodos de montagem do genoma. O objetivo geral de cada Assemblathon é fazer com que os grupos participantes tentem usar seu próprio *software* para cada um montar um ou mais

genomas que os organizadores do *Assemblathon* disponibilizam. A razão para organizar este *Assemblathon* é ver se podem ser produzidas métricas mais recentes para avaliar a qualidade de um conjunto de genomas que irão complementar as estatísticas existentes, tais como N50 e tamanho dos *contigs*. Os *Assemblathon* também existem pelo fato que existem muitos programas de montagem de genoma, mas nem sempre está claro como compará-los, já que não é fácil definir qual montador é melhor, ou qual pode funcionar bem em uma situação e em outras, ou como montar um genoma de alto conteúdo repetitivo.

3.2. Qualidade da Montagem

Existem pesquisas e ferramentas que propõem métodos para avaliar uma montagem. Neste caso temos o *QUAST* (GUREVICH et al., 2013) e o *REAPR* (HUNT; KIKUCHI; SANDERS, 2013).

O *QUAST* (GUREVICH et al., 2013) é uma ferramenta que tem a finalidade de gerar relatórios de análise de montagens de sequências de genomas. Ele tem a capacidade de analisar diversas montagens ao mesmo tempo, com ou sem a sequência de um organismo de referência. O *QUAST* permite uma extensa comparação de vários parâmetros como contagem e quantificação de nucleotídeos, propriedades do genoma, regiões de erros, inversões de montagem e regiões gênicas. O *QUAST* pode ser executado tendo genoma de referência ou não, com a única diferença que algumas estatísticas não podem ser calculadas quando o genoma de referência não é disponibilizado. Se tivermos o genoma de referência o *QUAST* pode ser executado de duas maneiras:

- Recebe como entrada o resultado da montagem, o genoma de referência, o arquivo de genes anotados, arquivo de operações.
- Recebe como entrada o resultado da montagem e o genoma de referência.

Se não tivermos o genoma de referência então o *QUAST* só recebe o resultado da montagem.

O *REAPR* (HUNT; KIKUCHI; SANDERS, 2013) identifica com precisão erros na montagem de genoma sem a necessidade de um genoma de referência. *REAPR* está validado em genomas completos, conjuntos de bactérias e *Caenorhabditis elegans*. Quando é aplicado a um projeto de genoma em andamento, o *REAPR*

fornece estatísticas de montagem corrigidas que permitem a comparação quantitativa de várias montagens. Para executar o REAPR não é preciso ter o genoma de referência, somente o resultado da montagem e o conjunto de *reads*.

A seguir listamos algumas estatísticas do *QUAST* e do *REAPR*:

- **Número *Contigs* (≥ x pb) N**úmero total de *contigs* de comprimento ≥ x pb
- **Comprimento total (≥ x pb)** Número total de bases em *contigs* de comprimento ≥ x pb
- **N50** (maior N tal que 50% do total de pares de bases do genoma esteja contida em contigs >= N pb).

  Por exemplo, considere 9 contigs com os comprimentos 2,3,4,5,6,7,8,9 e 10; sua soma é 54, metade da soma é de 27, e o tamanho do genoma também acontece de 54 pb. 50% dessa montagem seria 10 + 9 + 8 = 27 (metade do comprimento da seqüência). Assim, o N50 = 8, que é o tamanho do contig que, juntamente com os contigs maiores, contém metade da sequência de um genoma particular.
- **N60, N70, N75 N80 e N90** (Analogamente ao N50 só com percentual diferente 60%, 70%, 75%, 80 % e 90%)
- **NG50** (Comprimento para o qual a coleção de todos os *contigs* desse comprimento ou maior cobre pelo menos metade do genoma de referência. Esta métrica é calculada somente se o genoma de referência for fornecido.)
- **NG75** (São definidos de forma semelhante a NG50, mas com 75% em vez de 50%.)
- **L50** (Número de *contigs* igual ou maior que N50.) Dado um conjunto de contigs, cada um com seu próprio comprimento, a contagem de L50 é definida como o menor número de contigs cuja soma de comprimento produz N50. No exemplo de N50 acima, o L50 = 3.
- **L75** (Analogamente ao L50 só com percentual 75)
- **LG50** (Número de *contigs* igual ou maior que NG50.)
- **LG75** (Número de *contigs* igual ou maior que NG75.)
- **Alinhamento de maior comprimento.**
- **Comprimento total do alinhamento.**
- **Comprimento médio do alinhamento.**
- **Maior contig** (comprimento do maior contig na montagem.)

- **Comprimento total** (número total de bases na montagem.)
- **% Conteúdo GC**
- **Fração genômica** (Número total de bases alinhadas na referência dividido pelo tamanho do genoma.)
- **Proporção de duplicação** (Número total de bases alinhadas na montagem dividida pelo número total de bases alinhadas no genoma de referência.)
- **Bases livres de erros**

Como resultado do estudo dos trabalhos relacionados podemos concluir que os projetos e ferramentas têm propostas interessantes para a avaliação de montadores e as montagens produzidas por estes, mas ainda podem ser melhorados para obter uma solução mais completa. Nos dados oferecidos pelo projeto *GAGE* e o evento *Assemblathon* são baseados principalmente no valor de N50 e no tamanho dos *contigs* para avaliar a qualidade da montagem dado por um montador. Eles também não têm informação da influência do valor de *k* no resultado da montagem. As ferramentas *QUAST* e *REAPR* oferecem um conjunto de estatísticas para avaliar uma montagem, mas não conseguem fazer uma avaliação comparativa usando vários montadores. Nenhum dos trabalhos mencionados tem em consideração métricas que complementam a análise como: a quantidade de genes identificados, quantidade de genes identificados de bases corretas e genes identificados exatamente iguais na montagem a qual faz com que as análises dos resultados da montagem sejam mais completas e precisos.

Nosso trabalho cobre uma série de testes onde serão feitas comparações entre os montadores tendo em conta, além das métricas N50 e do tamanho dos *contigs,* análises de quantidade de genes corretamente identificados, análises do valor de *k* para o desempenho do montador e a qualidade da montagem. Também será desenvolvida uma ferramenta que vai juntar as estatísticas do *QUAST* e do *REAPR*, as análises de quantos e quais genes estão presentes na montagem e comparações de várias montagens usando todas as estatísticas.

No seguinte capítulo apresentaremos as características dos montadores usados para realizar os testes. Também são apresentadas as diferentes análises sobre as montagens.

4
Análise Computacional e Biológica

Sabemos que para selecionar um montador temos que ter em conta diferentes fatores ao mesmo tempo e chegar a um consenso de qual será o melhor a usar. A Seção 4.1 apresenta o grupo de montadores escolhidos despois da pesquisa nos trabalhos relacionados e da opinião de diferentes biólogos. A Seção 4.2 resume as análises computacionais e estatísticas feitas para avaliar a qualidade computacional desses montadores. Também resume os algoritmos que identificam corretamente os genes em uma montagem o que constitui uma das contribuições específicas desta dissertação.

4.1.
Montadores

Para realizar os experimentos foram escolhidos os montadores: Minia (CHIKHI; RIZK, 2012), Velvet (ZERBINO; BIRNEY, 2008), SOAPdenovo2 (LUO et al., 2012), SPAdes (BANKEVICH; NURK; ANTIPOV, 2012a) e MaSuRCA (ZIMIN et al., 2013). Já que são os mais representativos na literatura, além de ser utilizados em projetos de referências como *GAGE* (MAGOC et al., 2013; SALZBERG et al., 2012) e *Assemblathon* (EARL et al., 2011). Todos os montadores escolhidos são baseados no uso de grafos *de Bruijn* e realizam montagens *de novo.*

4.1.1.
Minia

O montador Minia (CHIKHI; RIZK, 2012) utiliza uma nova codificação para o grafo *de Bruijn* que permite realizar sua tarefa com a utilização de menos memória RAM e, teoricamente, com um tempo igual ou menor do que outros montadores do mesmo tipo.

A nova codificação do Minia baseia-se em uma estrutura de dados adicional que permite eliminar falsos positivos críticos do grafo *de Bruijn* original (CHIKHI; RIZK, 2012; SALIKHOV; SACOMOTO; KUCHEROV, 2013). Devido a isso, é possível simplificar ao máximo o consumo de memória RAM. Tendo menos erros em estruturas de dados faz com que o processo de montagem seja mais rápido. Minia produz resultados de similar contiguidade e precisão à outros montadores *de Bruijn* (por exemplo, Velvet) (CHIKHI; RIZK, 2012).

4.1.2. VELVET

O montador Velvet (ZERBINO; BIRNEY, 2008) pode montar qualquer tipo de *reads* mas, realmente, está concebido para a montagem de *reads* curtos que variam de 25 a 50 pb.

A principal vantagem deste montador é que divide em duas etapas independentes, mas complementares do processo de montagem. Essas etapas são executadas pelos programas Velveth e Velvetg. O Velveth é o responsável de formar a estrutura de dados utilizada para montar o genoma, eliminando erros produzidos pelo sequenciamento. O Velvetg é o núcleo do montador e é encarregado de resolver as repetições causadas pela complexidade do genoma.

O grafo *de Bruijn* usado no Velvet é constituído por blocos. Cada bloco é composto por um *k-mer* e pelo seu complemento reverso. O complemento reverso de um *k-mer* é obtido pelo reverso dele e o complemento de suas bases. Nós adjacentes possuem *k-mers* com sobreposição de *k-1* nucleotídeos. O grafo é construído utilizando uma tabela de *hash* dos *reads* indexados por um *k-mer*. Os *contigs* são produzidos pelo mapeamento dos *reads* e são formados seguindo as transições do grafo, unindo os que forem ambíguos (ZERBINO; BIRNEY, 2008).

4.1.3. SOAPdenovo2

É um novo método de montagem de short *reads* que pode construir um conjunto de montagem *de novo* para genomas de tamanho humano. O programa é especialmente concebido para montar *read*s obtidos mediante a tecnologia Illumina. O método constrói e realiza análises precisos de genomas inexplorados.

O *SOAPdenovo2* (LUO et al., 2012) é composto por módulos que lidam com a correção de erros dos *reads*, a construção do grafo *de Bruijn*, a montagem de *contigs*, a construção de *scaffold*s e fechamento de *gap*.

4.1.4. SPAdes

SPAdes (BANKEVICH; NURK; ANTIPOV, 2012b) é um dos montadores *de novo* que usam conjuntos de *reads* curtos como entrada. *SPAdes* usa os *k-mers* para construir o grafo inicial *de Bruijn*. Tem várias etapas nas quais executa operações baseadas na estrutura do grafo, a cobertura e o comprimento das sequências. Além disso, ele ajusta erros iterativamente (BANKEVICH; NURK; ANTIPOV, 2012a).

4.1.5. MaSuRCA

MaSuRCA (ZIMIN et al., 2013) é *software* de montagem de genoma completo. Ele combina a eficiência do grafo *de Bruijn* e *OLC*. Pode montar conjuntos de dados que contêm apenas short *reads* do sequenciamento Illumina ou uma mistura de short *reads* e long *reads* de *Sanger*, *454*, *PacBio* e *Nanopore*. É melhor para resolver estruturas repetitivas curtas.

4.2. Análise da Qualidade e Comparação

O processo de análise consiste em executar os montadores selecionados com genomas de bactérias de características diferentes e, em cada execução, variamos diferentes parâmetros dos montadores como o caso do valor de *k*, assim como os parâmetros específicos de cada um deles. Com as variações dos parâmetros buscamos, idealmente, obter os melhores resultados no menor tempo possível e com o menor consumo de recursos ou pelo menos ter informações para que o biólogo tome uma decisão baseada na prioridade e os recursos com que ele conta.

4.2.1.
Análise Computacional e Estatística

Nas pesquisas de eficiência computacional temos duas variáveis que são sempre testadas e comparadas: tempo de execução e memória. A variável do tempo de execução mede o tempo para obter uma resposta. A variável de memória mede o consumo da memória RAM.

A memória RAM é um aspecto crítico dos montadores já que eles fazem um uso exaustivo da memória RAM, tendo que armazenar *reads* e *k-mer*s para fazer comparações e, muitas vezes, os montadores não são capazes de dar um resultado pela falta de memória RAM para processamento. Temos então montadores que serão mais eficientes computacionalmente, mas que fazem um uso maior de memória RAM e, por vezes, não conseguem gerar um resultado.

É importante mencionar também que as variações nos parâmetros que os montadores usam podem ter um impacto considerável nos valores dessas variáveis. Por isso nós vamos a estudar nesta dissertação a influência do valor de *k* na qualidade da montagem. O tamanho do *k-mer* poder ter influência tanto computacionalmente como biologicamente.

Uma vez executadas todas as combinações possíveis de parâmetros extrairemos o consumo de memória RAM e tempo de execução. Usando as ferramentas *QUAST* (GUREVICH et al., 2013) e REAPR (HUNT; KIKUCHI; SANDERS, 2013) obtemos estatísticas sobre o resultado das montagens. No capítulo 3 podemos ver as estáticas que podem ser obtidas pelo *QUAST* e REAPR.

Uma das estatísticas mais importante é o N50, quanto maior for o valor do N50 para o conjunto obtido maior é o significado biológico da montagem, mas essa métrica pode não refletir com precisão a qualidade de uma montagem. Os cientistas concluíram que nenhum conjunto de métricas era perfeito (BAKER, 2012).

Com as estatísticas computacionais e as estatísticas da montagem dadas por *QUAST* e REAPR já temos um conjunto de métricas para comprar as montagens. Mesmo assim também é necessário fazer uma análise de quais e quantos genes estão corretamente codificados na montagem.

4.2.2.
Análise Biológica

Uma das análises que podem ser feitas para avaliar o resultado da montagem é identificar quantos e quais genes do genoma de referência estão presentes, e se os genes identificados têm as sequências de bases correta e o tamanho correto. Quando nos referimos a genes com tamanho correto estamos falando dos genes exatamente iguais. Para auxiliar nesta tarefa, foi proposto e desenvolvido nesta dissertação um programa automático para obter esses dados.

A análise é baseada num genoma de referência e o uso do programa *BLAST* (*Basic Local Alignment Search Tool*) ((US), 2008). O *BLAST* considera regiões de similaridade entre as sequências locais. O programa compara sequências de nucleotídeos ou proteínas com sequências de uma base de dados e calcula a significância estatística dos alinhamentos. O programa BLAST pode ser usado para inferir relações evolutivas e funcionais entre as sequências, assim como para ajudar a identificar os membros de famílias proteicas (ALTSCHULL et al., 1990).

Primeiramente mostraremos as sequências de ações feitas para obter as análises biológicas. Executamos usamos a aplicação *makeblastdb* ((US), 2008) que produz bancos de dados BLAST (MADDEN, 2013) a partir de arquivos FASTA, o banco é uma transformação das sequências com um formato especial que torna o processamento mais eficiente. Esse comando sempre tem que ser executado antes de executar o *blastn*; para isto e necessário ter o genoma de referência. Despois de executar o *makeblastdb* usamos *blastn* ((US), 2008) o qual faz o alinhamento entre o arquivo FASTA e o resultado do *makeblastdb*.

Depois de executar o *blastn* obtemos um arquivo. A **Figura 1** apresenta os dados que precisamos para fazer as análises biológicas. Para isto fazemos um processamento da informação obtida pelo *blastn*. Para as análises utilizamos a informação das colunas 2, 7, 8, 9, 10 e 11. A coluna 2 é o nome do gene, a coluna 7 registra a informação mais importante, já que se o valor for 0,0 significa que os genes são identificados corretamente. Nas colunas das 8 à 11 registra a informação dos comprimentos de cada gene. A **Figura 2** mostra o pseudocódigo da análise biológica.

No seguinte capítulo apresentaremos às configurações dos testes e os genomas de referências que foram utilizados. Também mostramos os resultados dos testes feitos.

Query id	Subject_id	%_identity	%_posittives	score_bit	score	e-value	q.start	q.end	s.start	s.end	q.length	s.length
scf7180000001138	AHA_0405	94.991	94.99	3289	6074	0.0	58128	62000	3870	1	121853	3870
scf7180000001138	AHA_0432	90.435	90.43	2624	4846	0.0	34309	37988	3680	1	121853	3684
scf7180000001138	AHA_0461	97.944	97.94	1917	3541	0.0	4043	6085	1	2043	121853	2043
scf7180000001138	AHA_0439	95.030	95.03	1695	3131	0.0	25681	27672	1992	1	121853	1992
scf7180000001138	AHA_0383	95.253	95.25	1680	3103	0.0	85383	87341	1959	1	121853	1959
scf7180000001138	AHA_0392	94.854	94.85	1446	2671	0.0	76125	77834	1710	1	121853	1710
scf7180000001138	AHA_0419	84.489	84.49	1416	2615	0.0	47966	50626	1	2667	121853	2667
scf7180000001138	AHA_0350	87.459	87.46	1319	2436	0.0	119734	121853	2535	416	121853	2553
scf7180000001138	AHA_0354	94.852	94.85	1314	2427	0.0	116832	118385	1	1554	121853	1554
scf7180000001138	AHA_0404	95.986	95.99	1293	2388	0.0	62031	63500	1470	1	121853	1470
scf7180000001138	AHA_0408	96.458	96.46	1287	2377	0.0	55324	56763	1440	1	121853	1440
scf7180000001138	AHA_0364	93.075	93.08	1266	2338	0.0	108447	110042	1602	1	121853	1602
scf7180000001138	AHA_0379	91.627	91.63	1252	2313	0.0	92343	94014	1681	10	121853	1710
scf7180000001138	AHA_0399	81.845	81.84	1225	2263	0.0	67919	70636	2721	1	121853	3684
scf7180000001138	AHA_0399	90.789	90.79	55	102	6.03e-20	66998	67073	3684	3609	121853	3684
scf7180000001138	AHA_0438	93.594	93.59	1197	2211	0.0	28157	29638	1482	1	121853	1482
scf7180000001138	AHA_0380	93.302	93.30	1192	2202	0.0	90786	92277	1533	42	121853	1533
scf7180000001138	AHA_0451	97.565	97.56	1180	2180	0.0	12104	13376	1273	1	121853	1275
scf7180000001138	AHA_0440	91.398	91.40	1171	2163	0.0	24105	25684	1578	1	121853	1617
scf7180000001138	AHA_0371	91.477	91.48	1143	2111	0.0	100985	102520	1902	367	121853	1902
scf7180000001138	AHA_0371	84.970	84.97	456	843	0.0	102165	103009	830	1	121853	1902
scf7180000001138	AHA_0371	83.465	83.46	253	468	5.16e-130	102101	102604	858	352	121853	1902
scf7180000001138	AHA_0371	84.464	84.46	244	451	5.20e-125	102093	102549	830	374	121853	1902
scf7180000001138	AHA_0371	83.801	83.80	235	435	5.23e-120	102055	102513	796	335	121853	1902
scf7180000001138	AHA_0371	82.979	82.98	224	414	6.82e-114	102066	102529	749	286	121853	1902
scf7180000001138	AHA_0371	84.659	84.66	189	350	1.95e-94	102239	102589	828	478	121853	1902
scf7180000001138	AHA_0371	86.319	86.32	181	335	5.46e-90	102063	102369	680	374	121853	1902
scf7180000001138	AHA_0358	95.530	95.53	1143	2111	0.0	113077	114396	1	1320	121853	1320
scf7180000001138	AHA_0463	97.119	97.12	1110	2050	0.0	1572	2786	1	1215	121853	1215
scf7180000001138	AHA_0377	94.943	94.94	1107	2045	0.0	95000	96304	1305	1	121853	1305
scf7180000001138	AHA_0393	96.151	96.15	1080	1995	0.0	74851	76071	1221	1	121853	1221
scf7180000001138	AHA_0431	94.366	94.37	1062	1962	0.0	38192	39469	1	1278	121853	1278

Figura 1. Exemplo da informação obtida pelo blastn

```
Por cada Montagem fazer o blastn
{
   quantidade_genes = 0;
   quantidade_genes_bases_corretas = 0;
   genes_exatamente_iguais = 0;
   list quant_genes = new list();

   result_blastn = Blastn();

   Por cada fila do result_blastn
   {
     if(quant_genes.NãoContem(Gene))
     {
       quant_genes.inserir(Gene));
       quantidade_genes ++;

       if(valor da coluna 7 igual a 0.0)
           quantidade_genes_bases_corretas ++;

       if(Gene tem o mesmo comprimento)// iguais se resta da coluna 8 e 9 é igual a resta de 10 e 11
           genes_exatamente_iguais ++;
     }

     else
     {
       if(Gene armazenado valor da coluna 7 não e igual a 0.0 e
       Gene.valor_da_coluna 7 == 0.0)
       {
           Então tracamos informação de Gene e quantidade_genes_bases_corretas ++;

           if(Gene tem o mesmo comprimento)
               genes_exatamente_iguais ++;
       }
     }
   }
}
```

Figura 2. Pseudocódigo da análise biológica

Agora explicaremos brevemente o Pseudocódigo. Este método é executado para cada montagem. Podemos ver na **Figura 2** que as linhas do código estão numeradas. Nas linhas da 3 a 7 temos a criação de variáveis que utilizamos, na linha 8 executamos o *blastn*. Nas linhas da 10 a 36 temos a análise feito por cada linha do arquivo resultado do *blastn.* Nas linhas da 12 a 22 fazemos a análise se não temos armazenado o gene identificado. Na linha 14 armazenamos o gene e na linha 15 aumentamos a quantidade de genes encontrados até o momento. Nas linhas 17 e 18 verificamos se o gene identificado tem as sequências de bases corretas. Nas linhas 20 e 21 verificamos se o gene tem o mesmo comprimento. Nas linhas da 26 a 33 fazemos a análise se já temos o gene armazenado. Nas linhas da 27 a 33 verificamos se o gene tem as sequências de bases corretas além do mesmo comprimento.

5
Testes e Resultados

Este capítulo apresenta os resultados dos testes realizados com os diferentes montadores. A finalidade dos testes é entender a influência de alguns parâmetros na execução dos montadores e as possíveis correlações com qualidade dos resultados obtidos na montagem. O biólogo terá em mãos um estudo que permitirá fazer escolhas. A Seção 6.1 especifica as configurações dos testes e Seção 6.2 mostra os resultados obtidos.

5.1.
Configuração dos Testes

O processo consiste em executar os montadores selecionados, Minia (CHIKHI; RIZK, 2012), Velvet (ZERBINO; BIRNEY, 2008), SOAPdenovo2 (LUO et al., 2012), SPAdes (BANKEVICH; NURK; ANTIPOV, 2012a) e MaSuRCA (ZIMIN et al., 2013) com oito *dataset* de características diferentes. Estes *dataset* correspondem a genomas de bactérias disponibilizados no GAGE-B (MAGOC et al., 2013). A Tabela 1 apresenta um resumo das características desses genomas e as estatísticas do sequenciamento e qualidade dos *reads*.

Nessa tabela podemos ver umas das principais características temos o tamanho de cada genoma assim como a quantidade de genes que contêm cada um. Também temos o comprimento dos *reads* e dos fragmentos.

Os genomas de referência, os arquivos de genes anotados e os arquivos de operações desses genomas foram obtidos no site da NCBI[1], segue os nomes dos genomas correspondentes no site da NCBI:

DS1. *Aeromonas hydrophila* hydrophila ATCC 7966

DS2. *Bacillus cereus* ATCC 14579

DS3. *Bacteroides fragilis* YCH46

[1] https://www.ncbi.nlm.nih.gov

DS4. *Mycobacterium abscessus* ATCC 19977

DS5. *Rhodobacter sphaeroides* 2.4.1

DS6. *Staphylococcus aureus* aureus NCTC 8325

DS7. *Vibrio cholerae* O1 biovar El Tor str. N16961

DS8. *Xanthomonas axonopodis* pv. citrumelo F1

Tabela 1. Características dos Genomas Escolhidos

Bactéria	Tamanho genoma(MB)	Conteúdo GC (%)	Comprimento *reads* (bp)	Quantidade de genes	Cobertura
Aeromonas hydrophila subsp. hydrophila ATCC 7966	4,74	61,5	101	4284	250x
Bacillus cereus ATCC 14579	5,41	35,3	101	5494	100–300×
Bacteroides fragilis YCH46	5,28	43,3	101	4717	250x
Mycobacterium abscessus ATCC 19977	5,07	64,1	100	4992	115x
Rhodobacter sphaeroides 2.4.1	4,6	69	101	4474	210x
Staphylococcus aureus subsp. aureus NCTC 8325	2,82	32,9	101	2872	250x
Vibrio cholerae O1 biovar El Tor str. N16961	4,0	47,7	100	3693	110x
Xanthomonas axonopodis pv. citrumelo F1	4,97	64,9	101	4376	250x

Cada um dos montadores utilizados precisa para sua execução definir o parâmetro *k*, correspondente ao tamanho do *k-mer*. A ideia é testar cada montador com mais de um valor de *k*. Foram usados os valores de *k* entre 23 e 95 já que isto permite testar um conjunto grande de valores de *k*, este conjunto tem valores bem menores, médios e quase iguais ao comprimento do *read*. Para o plano de testes foram usados 37 valores de *k*, entre 23 e 95. Como temos oito dataset, cinco montadores e 37 valores de *k* então, a quantidade de montagens analisadas foram: $8 * 5 * 37 = 1480$.

Todos os testes foram feitos no mesmo computador, que conta com 2 Processadores Intel (R) Xeon (R) CPU E5-2620 0 @ 2.00GHz (6 core), 32 GB de memória RAM e 2 HDS (500 GB + 1TB).

Para cada montador foi necessário também ajustar parâmetros específicos. Cada montador tem um manual onde se especificam as configurações.

- **Minia** *(http://minia.genouest.org/files/manual.pdf*)
- **Velvet** (*http://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf*)
- **SOAPdenovo2** (http://soap.genomics.org.cn/soapdenovo.html)
- **SPAdes**(http://spades.bioinf.spbau.ru/release3.10.0/manual.html)
- **MaSuRCA**(ftp://ftp.genome.umd.edu/pub/MaSuRCA/MaSuRCA_QuickStartGuide.pdf)

5.2. Resultados

Para analisar os resultados obtidos com todas as execuções e com os diferentes parâmetros de entrada para cada montador, vamos apresentar neste capítulo um estudo para análise das informações e impacto de cada parâmetro. Para cada análise de um parâmetro apresentaremos uma tabela de *ranking* com a ordem obtida. No caso, o melhor *ranking* 1 e o pior 5, dos montadores por genoma. O *ranking* foi feito tendo em conta a média dos 10 melhores resultados do montador. Os valores que vamos analisar nesta seção são os seguintes:

- Tempo de Execução e consumo de Memória RAM.
- Número de *Contigs* e valores N50.
- Quantidade de Genes Identificados, de Genes identificados com sequências de bases corretas e de Genes Identificados exatamente iguais.

Esses valores serão representados para cada valor *k* utilizado nos testes.

5.2.1. Resultados do Tempo de Execução dos experimentos

Com relação ao tempo de execução dos montadores com todos os *dataset* podemos observar na Figura 3 que o comportamento de cada montador é bastante semelhante, eles diminuem o tempo na medida que aumenta o valor de *k*.

Um olhar atento nos gráficos permite observar que para os maiores valores de *k* obtemos os melhores resultados exceto para os genomas *Vibrio cholerae*, *Mycobacterium abscessus* e *Rhodobacter sphaeroides*. Esse é o comportamento esperado já que se diminui a quantidades de *k-mer* a ser analisados pelos montadores. Além disso vemos que o conteúdo de GC não gera um comportamento diferente. Na Tabela 2 temos *ranking* dos montadores, o melhor montador em todos nossos testes é o Minia. O SPAdes sempre leva mais tempo que todos os demais montadores.

Tabela 2. *Ranking* de Tempo por Genoma

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	4	1	2	5	3
Bacillus cereus	4	1	3	5	2
Bacteroides fragilis	4	1	2	5	3
Mycobacterium abscessus	4	1	3	5	2
Rhodobacter sphaeroides	4	1	3	5	2
Staphylococcus_aureus	4	1	2	5	3
Vibrio cholerae	4	1	3	5	2
Xanthomonas axonopodis	4	1	3	5	2
MELHOR MONTADOR	4	1	2.63	5	2.38

5.2.2. Consumo de Memória RAM dos experimentos

Outra análise importante é o consumo de recursos utilizado por cada montador. Neste caso analisaremos o consumo de memória RAM. Podemos ver na Figura 4 que todas os gráficos têm quase o mesmo comportamento independentemente dos valores de GC. O MaSuRCA mante-se quase constante no consumo e o Minia e o Velvet tendem a diminuir o consumo de memória na medida que aumenta o valor de *k*.

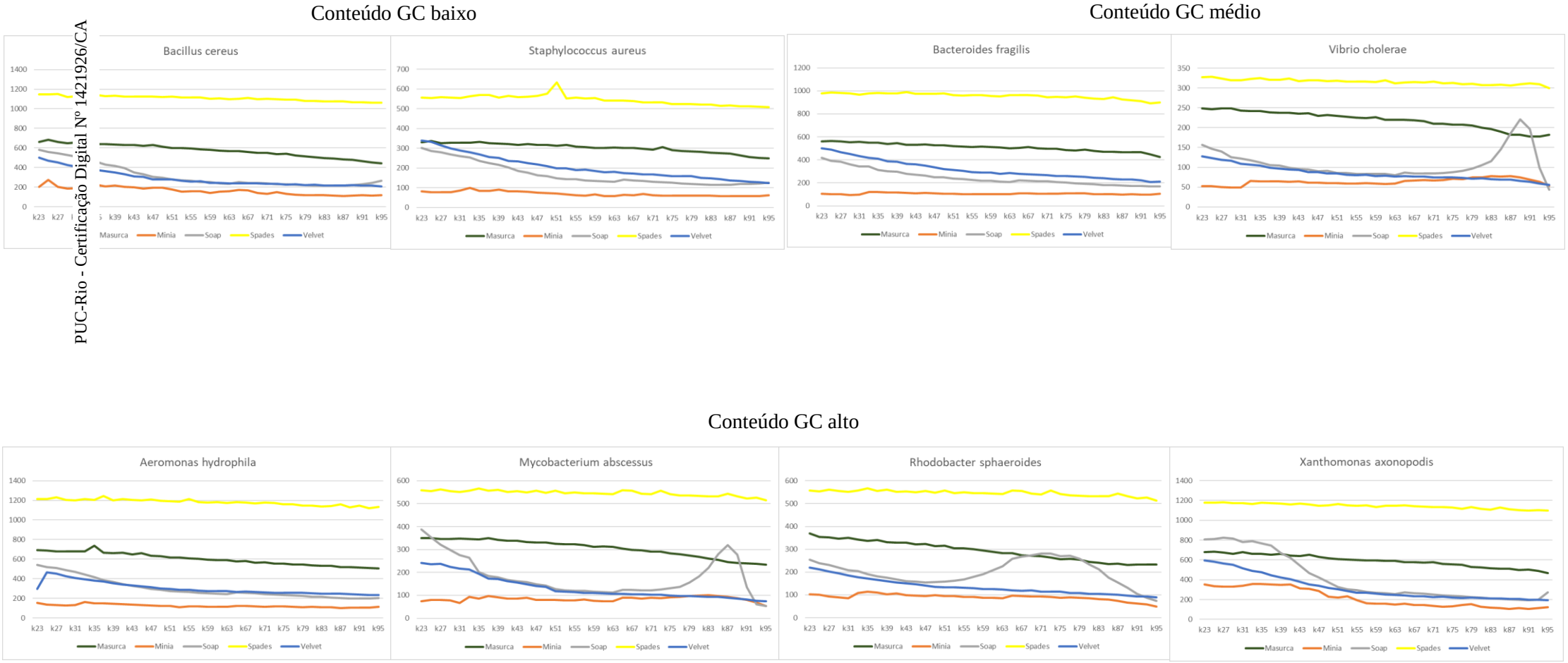


Figura 3. Tempo de execução de cada montador com os *dataset*

Observando os gráficos podemos perceber também que para os maiores *k* obtemos os melhores resultados com exceção do SOAP.

Os montadores SOAP e SPAdes são os montadores que têm o maior consumo de memória RAM e o Minia o que menos consume, podemos observar na Tabela 3.

Se nesse momento tivéssemos que escolher o melhor montador sem dúvida alguma seria o Mina já que foi o de melhor desempenho com relação ao tempo e ao consumo de memória. E se tivéssemos que selecionar um *k* para fazer nossas análises seria escolhido um dos valores mais altos.

Tabela 3. *Ranking* de Memória por Genoma

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	2	1	4	5	3
Bacillus cereus	2	1	5	4	3
Bacteroides fragilis	2	1	4	5	3
Mycobacterium abscessus	2	1	5	4	3
Rhodobacter sphaeroides	2	1	5	4	3
Staphylococcus aureus	2	1	5	4	3
Vibrio cholerae	3	1	5	4	2
Xanthomonas axonopodis	2	1	5	4	3
MONTADOR MELHOR	2.13	1	4.75	4.55	2.88

5.2.3. Número de *Contigs* dos experimentos

Uma análise muito importante é o número de *contigs* identificados na montagem. Quanto menor for o número de *contigs* melhor tende a ser a montagem. Na Figura 5 observamos o comportamento de cada montador com todos os *dataset*.

Nos gráficos com conteúdo de GC baixo podemos ver que o comportamento é bem similar em os dois gráficos. Eles começam diminuindo o número de *contigs* na medida que aumenta o valor de *k*, logo mantém-se quase constante até que nos últimos valores de *k* têm um aumento na medida que aumenta *k*.

Nos gráficos com conteúdo de GC médio podemos ver que no gráfico do genoma *Bacteroides fragilis* o comportamento é bem similar aos gráficos de conteúdo de GC baixo com exceção do montador MaSuRCA onde em todo momento a medida

que aumenta o valor de *k* diminui o número de *contigs*. No genoma *Vibrio cholerae* na medida que o valor de *k* aumenta, o número de *contigs* também aumenta até um ponto onde diminui o número de *contigs* identificados com exceção do montador MaSuRCA onde em todo momento a medida que aumenta o valor de *k* diminui o número de *contigs* até os valores finais de *k* onde dá um pequeno aumento no número de *contigs.*

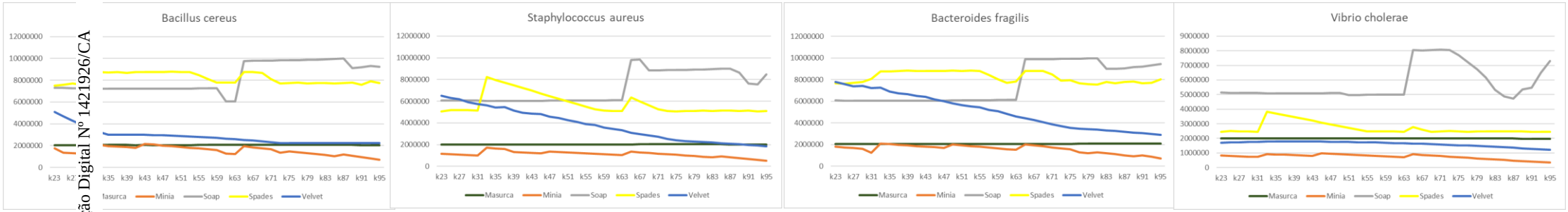
Nos gráficos com conteúdo de GC alto podemos ver vários comportamentos. No genoma *Aeromonas hydrophila* tem comportamento similar ao *Bacteroides fragilis.* No genoma *Mycobacterium abscessus* tem comportamento similar ao *Vibrio cholerae*. No genoma *Rhodobacter sphaeroides* na medida que o valor de *k* aumenta, o número de *contigs* também aumenta até um ponto onde diminui o número de *contigs* identificados com exceção do montador MaSuRCA onde em todo momento a medida que aumenta o valor de *k* aumenta o número de *contigs* até os valores finais de *k* onde diminui um pouco o número de *contigs.* No genoma *Xanthomonas axonopodis* tem comportamento similar aos genomas de conteúdo de GC baixo.

Em cinco dos oito *dataset* quatro montadores aumentam o número de *contigs* nos valores finais de *k*. Em uns casos percebemos que os melhores resultados geralmente são obtidos nos valores médios de *k*. Além disso podemos ver que em alguns casos obtêm-se bons resultados nos valores finais de *k*.

Os montadores de melhores desempenho são o SOAP e SPAdes. Eles quase sempre são os de melhores resultados enquanto a número de *contigs* mais em nossos experimentos pode-se apreciar que o melhor é o SOAP, o que é confirmado na média de *rankings* da Tabela 4. O montador de pior desempenho é o Minia, o qual em quase todos os testes identifica o maior número de *contigs*.

Conteúdo GC baixo

Conteúdo GC médio



Conteúdo GC alto

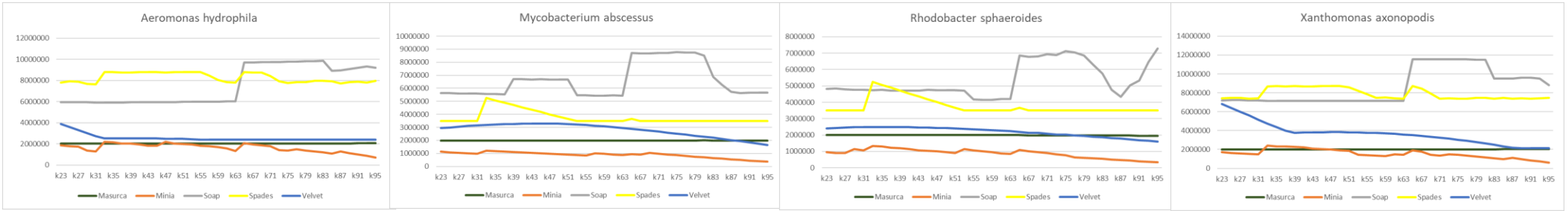


Figura 4. Consumo de memória de cada montador com os dataset

Tabela 4. *Ranking* de Número de *Contigs* por Genoma

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	4	5	1	2	3
Bacillus cereus	1	5	2	3	4
Bacteroides fragilis	5	4	1	2	3
Mycobacterium abscessus	5	4	1	2	3
Rhodobacter sphaeroides	2	5	1	3	4
Staphylococcus aureus	1	5	2	3	4
Vibrio cholerae	4	5	1	2	3
Xanthomonas axonopodis	2	4	5	1	3
MONTADOR MELHOR	3	4.63	1.75	2.38	3.38

Por tanto, ao realizar uma análise de qual seria o melhor montador considerando só o número de *contigs*, concluímos que é o SOAP.

Comparando com os resultados de tempo e memória vemos que o SOAP é bastante bom em tempo de execução é o pior enquanto ao consumo de memória. Mas o Minia que foi escolhido como melhor agora está entre os dois de pior desempenho.

5.2.4. N50

O parâmetro estatístico N50 tem uma importância para as análises de saber qual é o montador de melhor resultado. Quanto maior for o valor de N50 melhor tende a ser a montagem. Na Figura 6 podemos ver que os montadores apresentam um comportamento similar.

Nos gráficos com conteúdo de GC baixo podemos ver que o comportamento é bem similar em os dois gráficos. Eles começam aumentando o valor de N50 na medida que aumenta o valor de *k*, logo nos últimos valores de *k* desmuniu valor de N50 na medida que aumenta *k*. Com exceção do montador MaSuRCA que no caso do genoma *Bacillus cereus* se matem quase constante até os valores finais de *k* onde diminui o valor de N50 e no caso do *Staphylococcus aureus* aumenta em todo momento o N50.

Nos gráficos com conteúdo de GC médio podemos ver que o gráfico do genoma *Bacteroides fragilis* o comportamento é bem similar ao gráfico do *Staphylococcus aureus*. No genoma *Vibrio cholerae* na medida que o valor de *k* aumenta o valor de N50 diminui, com exação do montador SOAP que no primer momento aumenta o valor de N50 e depois diminui.

Nos gráficos com conteúdo de GC alto podemos ver vários comportamentos. No genoma *Aeromonas hydrophila* tem comportamento similar ao *Staphylococcus aureus.* No genoma *Mycobacterium abscessus* tem comportamento similar ao *Bacillus cereus*. No genoma *Rhodobacter sphaeroides* tem comportamento similar ao *Vibrio cholerae.* No genoma *Xanthomonas axonopodis* tem comportamento similar ao *Bacteroides fragilis*.

Em resumo, nos oito genomas temos que pelo menos quatro montadores diminuem o valor de N50 nos valores finais de *k*. No *dataset Bacillus cereus* os cinco montadores apresentam os piores resultados nos valores finais de *k.*

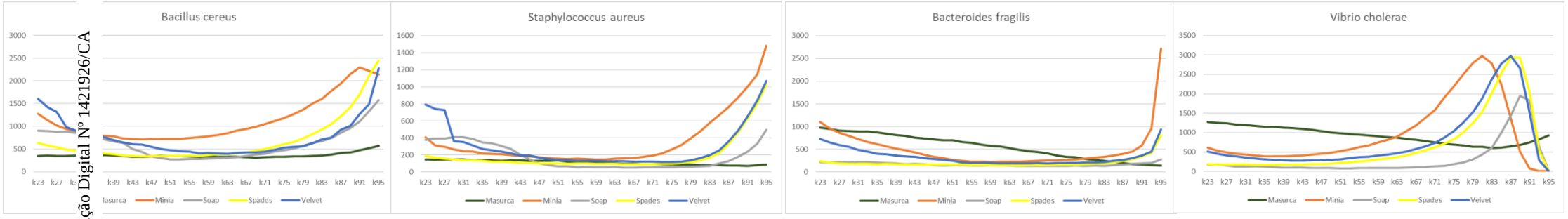
Na Tabela 5, observamos que, atendendo aos *rankings* dos montadores por genoma, SOAP e SPAdes são os de melhores resultados. Mas o SOAP em todos os experimentos feitos foi o de melhor desempenho, isto pode ser visto com mais detalhes na Figura 6. O Minia é o montador que obtêm menor valor de N50.

Tabela 5. *Ranking* de N50 por Genoma

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	4	5	2	1	3
Bacillus cereus	1	5	3	2	4
Bacteroides fragilis	5	4	1	2	3
Mycobacterium abscessus	5	4	1	2	3
Rhodobacter sphaeroides	3	5	1	2	4
Staphylococcus aureus	2	5	1	3	4
Vibrio cholerae	5	4	1	2	3
Xanthomonas axonopodis	4	5	2	1	3
MONTADOR MELHOR	3.63	4.63	1.5	1.88	3.38

Conteúdo GC baixo

Conteúdo GC médio



Conteúdo GC alto

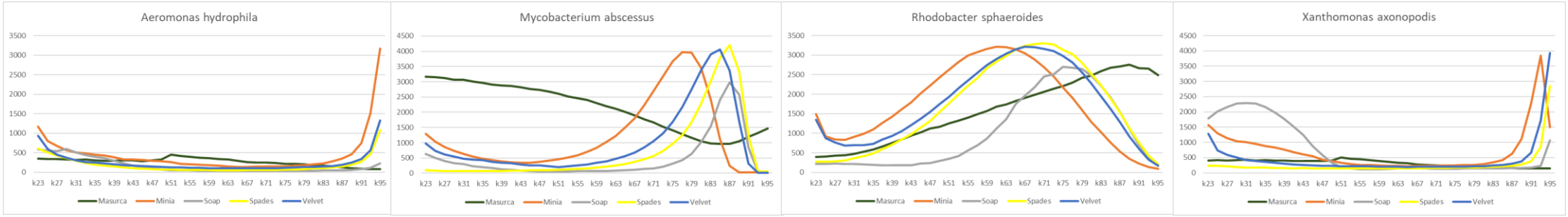
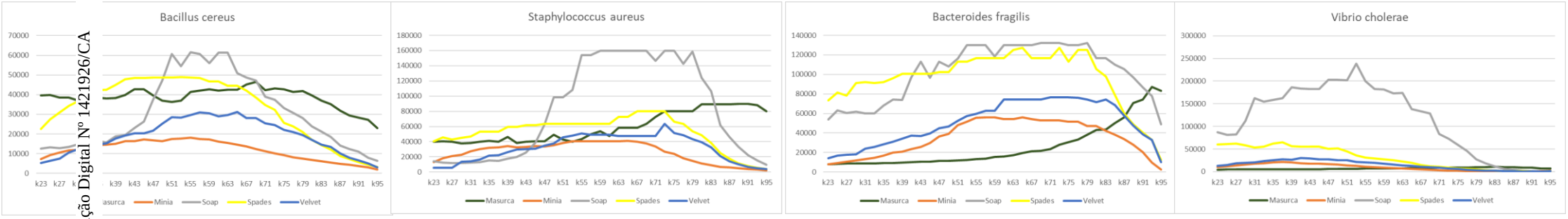


Figura 5. Número de contigs identificados por cada montador com os *dataset*

Conteúdo GC baixo

Conteúdo GC médio



Conteúdo GC alto

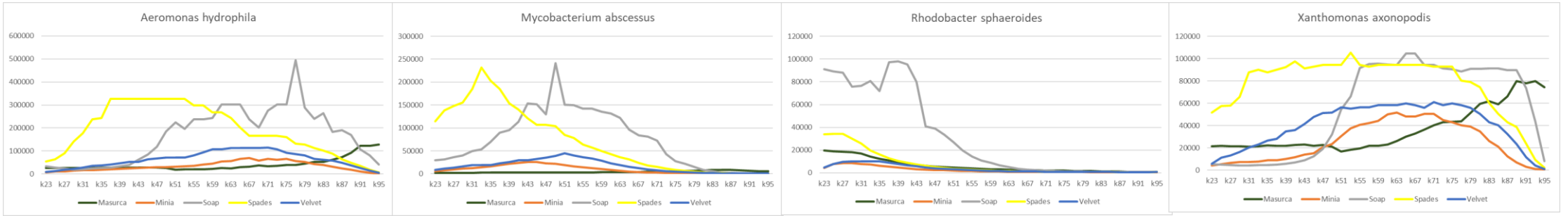


Figura 6. Valores de N50 identificados por cada montador com os *dataset*

Ao fazer uma análise de qual seria o melhor montador atendendo só ao valor de N50, então o SOAP seria o escolhido. Comparando com os resultados anteriores temos que o SOAP é o de melhor desempenho enquanto ao número de *contigs* e ao valor de N50. No entanto, o Minia está entre os de pior desempenho no cálculo desse parâmetro.

5.2.5. Quantidade de Genes Identificados

Com o parâmetro de quantidade de genes identificados começamos a analisar os parâmetros biológicos os quais são os de maior importância neste trabalho. Quanto maior é o número de genes identificados corretamente melhor vai ser a montagem. Na Figura 7 observamos o comportamento de cada montador com todos os *dataset*.

Nos gráficos com conteúdo de GC baixo podemos ver que o comportamento é bem similar, cada montador identifica quase a mesma quantidade de genes independentemente o valor de *k*. O SOAP identifica a maior quantidade de genes.

Nos gráficos com conteúdo de GC médio podemos ver que o gráfico do genoma *Bacteroides fragilis* o comportamento é bem similar aos gráficos de conteúdo de GC baixo. No genoma *Vibrio cholerae* acontece o mesmo comportamento com exceção dos montadores Velvet e o MaSuRCA, que na medida que o valor de *k* aumenta, a quantidade de genes também aumenta até um ponto onde se mantém quase constante.

Nos gráficos com conteúdo de GC alto podemos ver que o comportamento é bem similar exceto no genoma *Mycobacterium abscessus*, que na medida que o valor de *k* aumenta, a quantidade de genes também aumenta até um ponto onde se mantém quase constante.

O SOAP é o montador que identifica a maior quantidade de genes enquanto os montadores que menos identificam são o Velvet e o MaSuRCA. A Tabela 6 mostra o *ranking* de montadores por cada genoma.

Tabela 6. *Ranking* de Genes Identificados

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	3	3	1	2	3
Bacillus cereus	3	3	1	2	3

Bacteroides fragilis	3	3	1	2	3
Mycobacterium abscessus	2	2	1	2	2
Rhodobacter sphaeroides	3	3	1	2	3
Staphylococcus_aureus	5	3	1	2	4
Vibrio cholerae	5	2	1	3	4
Xanthomonas axonopodis	3	3	1	2	3
MONTADOR MELHOR RE	3.375	2.75	1	2.125	3.125

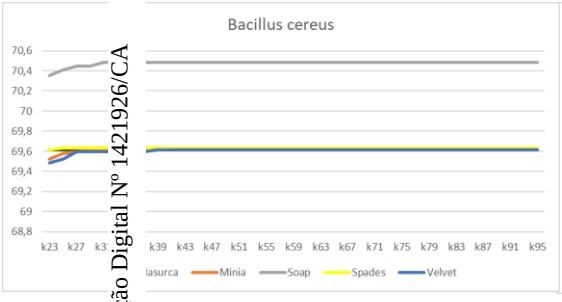
Ao analisar qual seria o montador de melhor desempenho, considerando só a quantidade de genes identificados, concluímos que é o SOAP. Comparando com os resultados anteriores temos que o SOAP é o de melhor resultado enquanto ao número de *contigs* e valor de N50.

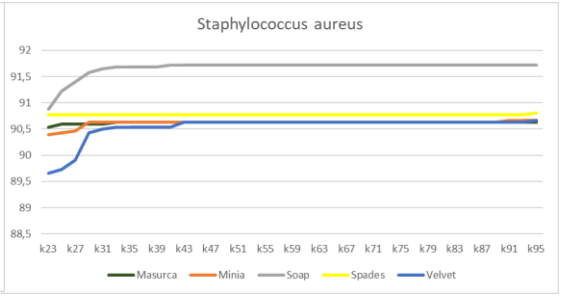
5.2.6. Genes Identificados com Bases Corretas

Outro parâmetro biológico é a quantidade de genes identificados com as bases corretas. Este parâmetro é tão importante quanto o anterior, pois constitui um refinamento da análise anterior, já que não é só identificar genes senão fazê-lo corretamente. Assim que, quanto maior é o número de genes com bases corretas melhor será o resultado da montagem. Na Figura 8 apresentamos como é o comportamento de cada montador com todos os *dataset*. Na maioria dos gráficos na medida que aumenta o valor de *k*, a quantidade de genes em ocasiões também aumenta ou se mantém quase constante, logo a partir de um valor de *k* começa a diminuir. Podemos observar que pelo menos dois montadores em todos os *dataset* diminuem os genes identificados com base correta nos valores finais de *k*. No *dataset Rhodobacter sphaeroides* os cinco montadores apresentam piores resultados nos valores finais de *k*.

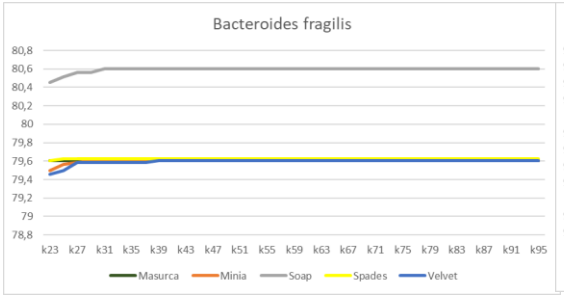
O montador de melhor desempenho é o SPAdes, o qual identifica a maior quantidade de genes com bases corretas, e o MaSuRCA é o de pior resultado ver na Tabela 7.

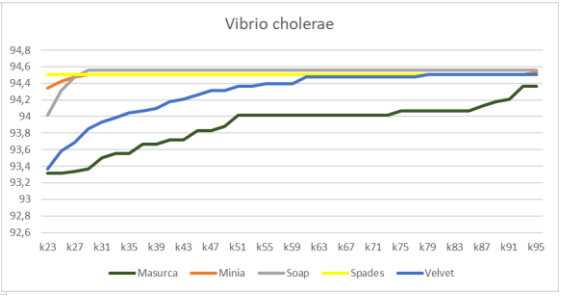
Conteúdo GC baixo



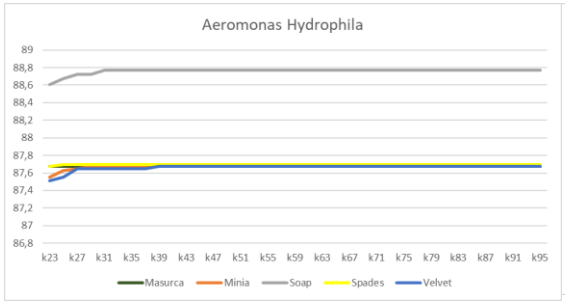


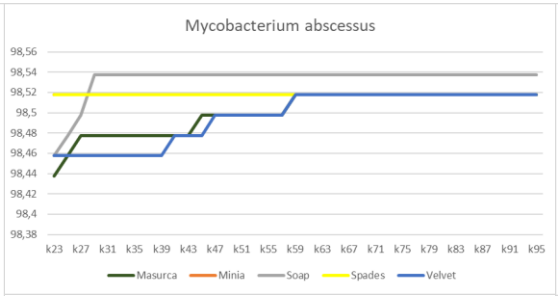
Conteúdo GC médio

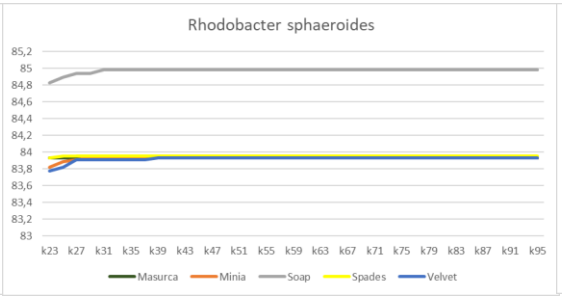




Conteúdo GC alto







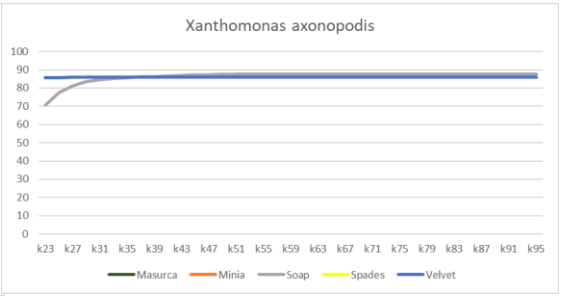


Figura 7. Quantidade genes identificados de cada montador com os *dataset*

Tabela 7. *Ranking* por Genes Identificados com Base Correta

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	5	2	4	3	1
Bacillus cereus	4	3	2	1	5
Bacteroides fragilis	5	2	3	1	4
Mycobacterium abscessus	4	2	3	1	2
Rhodobacter sphaeroides	5	4	1	2	3
Staphylococcus_aureus	2	4	1	3	5
Vibrio cholerae	5	2	3	1	4
Xanthomonas axonopodis	3	4	5	1	2
MONTADOR MELHOR	4.125	2.875	2.75	1.625	3.25

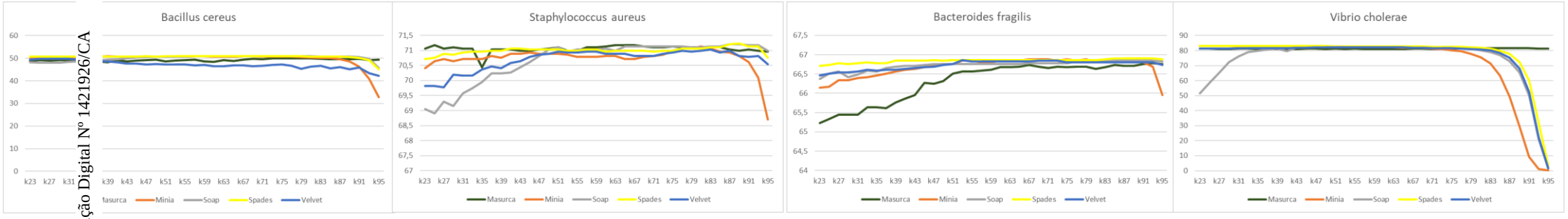
5.2.7. Genes Identificados Exatamente Iguais

Outro parâmetro biológico a analisar é a quantidade de genes identificados exatamente iguais (bases corretas e mesmo tamanho). Este parâmetro é o mais importante de todos. Assim que, quanto maior seja a quantidade de genes o resultado da montagem será muito melhor. A Figura 9 ilustra o comportamento de cada montador com os *dataset*. O comportamento neste parâmetro é muito parecido com o parâmetro anterior, na medida que aumenta o valor de *k* a quantidade de genes em ocasiones aumenta o se mantém quase constante, e a partir de um valor de *k* quase no final começa a diminuir. Como acontece no parâmetro anterior temos que pelo menos dois montadores em todos os *dataset* diminuem os genes identificados iguais nos valores finais de *k*. No *dataset Rhodobacter sphaeroides*, os cinco montadores obtêm os piores resultados nos valores finais de *k*.

Igualmente que no caso do parâmetro anterior, o montador de melhor desempenho é o SPAdes, o qual identifica a maior quantidade de genes exatamente iguais, e o MaSuRCA é o de piores resultado (ver Tabela 8).

Conteúdo GC baixo

Conteúdo GC médio



Conteúdo GC alto

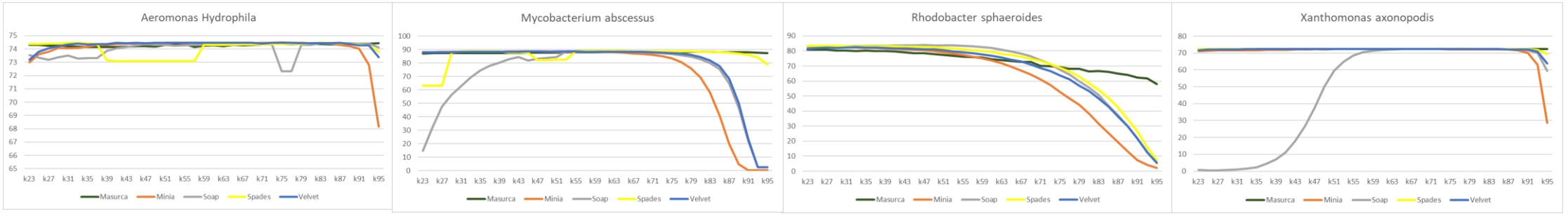


Figura 8. Quantidade genes identificados com bases corretas por cada montador com os *dataset*

Tabela 8. *Ranking* de Genomas identificados Exatamente Iguais

Genomas/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Aeromonas hydrophila	4	1	5	3	2
Bacillus cereus	4	3	3	1	5
Bacteroides fragilis	5	3	4	1	2
Mycobacterium abscessus	5	2	4	1	3
Rhodobacter sphaeroides	5	4	1	2	3
Staphylococcus aureus	2	1	4	3	5
Vibrio cholerae	5	2	3	1	4
Xanthomonas axonopodis	2	4	3	1	5
MONTADOR MELHOR	4	2.5	3.375	1.625	3.625

5.2.8. Montador de melhores resultados

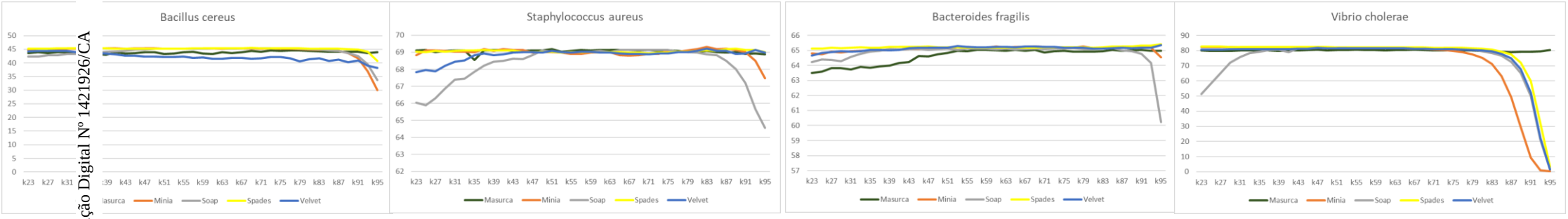
A partir das análises feitas com cada um dos parâmetros, observamos que montadores que são melhores em alguma análise não são tão bons em outras. Também sabemos que existem análises que são mais importantes para o biólogo na hora de escolher uma montagem. Assim que, para determinar o montador de melhor desempenho fizemos uma média ponderada dos *rankings* mostrados nas Tabela 2, Tabela 3, Tabela 4, Tabela 5, Tabela 6, Tabela 7 e Tabela 8, considerando que as análises biológicas têm um peso maior do que as computacionais, principalmente na hora de identificar os genes corretamente. Com isso tentamos garantir que o montador escolhido seja muito bom identificando os genes, mas que também não seja ruim nos demais parâmetros. Os valores dos pesos assignados foram determinados por nós.

Os pesos assignados a cada análise para o cálculo do melhor montador são:

- Tempo de Execução: 0,5
- Memória: 0,7
- Número de *Contigs*: 1,5
- N50: 2
- Genes: 3
- Genes com Sequência de Base Correta: 4
- Genes com Tamanho Correto: 5

Conteúdo GC baixo

Conteúdo GC médio



Conteúdo GC alto

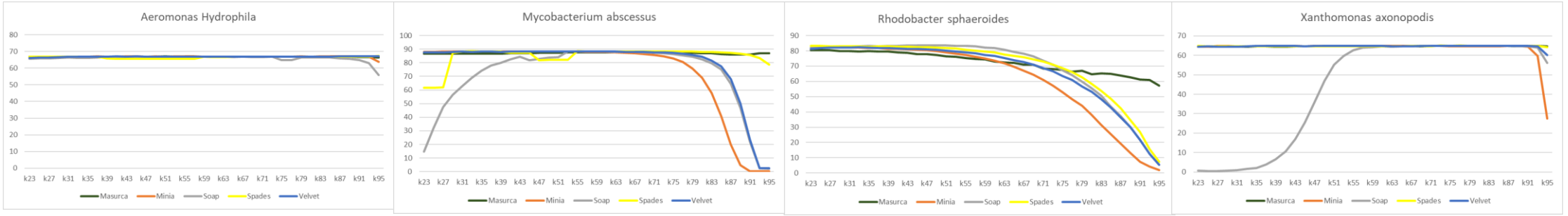


Figura 9. Quantidade genes identificados exatamente iguais por cada montador com os *dataset*

A Tabela 9 mostra o melhor montador segundo os pesos dados em cada análise. O melhor montador resultou ser o SPAdes. Esse montador é o melhor identificando os genes corretamente, tem bons resultados em N50 e número de *contigs*. Também podemos ver que é o pior montador só em tempo de execução.

Tabela 9. Melhor Montador baseado em uma Média Ponderada de *rankings*

Análise/Montador	MaSuRCA	Minia	SOAP	SPAdes	Velvet
Tempo de Execução	4	1	2.63	5	2.38
Memória	2.13	1	4.75	4.55	2.88
Número de *Contigs*	3	4.63	1.75	2.38	3.38
N50	3.63	4.63	1.5	1.88	3.38
Genes Identificados	3.375	2.75	1	2.125	3.125
Genes com Sequência de Base Correta	4.125	2.875	2.75	1.625	3.25
Genes exatamente iguais	4	2.5	3.375	1.625	3.625
Média Ponderada	**3.705**	**2.973**	**2.463**	**2.036**	**3.325**

5.2.9. Melhor Valor de *k*

Na maioria dos montadores precisamos de um valor inteiro *k* correspondente ao tamanho do *k-mer*, pois um *k-mer* errado pode influir no rendimento computacional (PÉREZ; GUTIERREZ; VERA, 2016) e na qualidade da montagem final. A ferramenta *KmerGenie* propõe a análise dos dados a priori para recomendar o comprimento do *k-mer*. Alguns testes mostram que as escolhas da *KmerGenie* levam a montagens próximas das melhores possíveis em todos os comprimentos do *k-mer* (CHIKHI; MEDVEDEV, 2013).

Na Tabela 10 mostramos os valores de *k* recomendados. Além disso, apresentamos o valor de *k* escolhido como melhor valor para os testes feitos.

Tabela 10. Valores de *k* recomendados pela Ferramenta *KmerGenie* e o escolhido

FERRAMENTA/ DATASET	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7	DS 8
GAGE	31	31	31	31	31	31	31	31
KmerGenie	31	49	89	31	23	31	47	85

k escolhido	33	69	87	33	23	43	27	77

Na Tabela 11 podemos ver os valores dos parâmetros que analisamos para escolher o melhor *k* dos testes feitos com o montador SPAdes e os oito *dataset*. A coluna sombreada tem os resultados do valor de *k* recomendado por *KmerGenie*, a outra coluna os valores de *k* escolhidos pela nossa análise como o melhor, e valor em negrito é o melhor valor. Nas colunas de quantidade de genes identificado (# genes), genes com bases corretas (G. Bases) e genes exatamente iguais (G. Iguais) estão representadas a porcentagem dos genes identificados. A quinta fila corresponde ao quinto *dataset* só tem os valores do *k* recomendado *KmerGenie* já que coincide com o nosso valor de *k* escolhido. Podemos observar que com os valores de *k* escolhidos pela nossa análise obtêm-se melhores resultados em quase todos os *dataset*. Portanto, concluímos que não podemos assegurar em que ferramenta *KmerGenie* recomende o melhor *k* para obter os melhores resultados.

Tabela 11. Comparação dos valores de *k* recomendado no Montador SPAdes. *K* recomendado contra *k* escolhido

Dataset	# *Contigs*		N50		# Genes		G. Bases		G. Iguais	
DS1	295	**242**	177768	**237501**	87,69	87,69	74,45	74,45	66,89	66,89
DS2	**344**	459	**48796**	38540	69,63	69,63	50,79	**50,94**	45,27	**45,45**
DS3	281	**245**	48932	**60228**	79,62	79,62	66,90	66,90	65,27	65,27
DS4	73	**69**	185491	**231741**	98,51	98,51	88,76	**88,80**	88,44	**88,46**
DS5	274		33794		83,92		83,39		83,19	
DS6	139	**108**	46567	**61969**	90,77	90,77	70,92	**71,06**	69,08	**69,11**
DS7	192	**182**	50793	**62071**	94,50	94,50	82,94	**83,15**	82,34	**82,56**
DS8	224	**179**	50384	**80179**	85,83	85,83	72,39	**72,69**	64,89	**65,05**

6
Aplicação

Neste capítulo são explicados alguns detalhes do desenvolvimento da uma aplicação integradora. Essa aplicação permite calcular as estatísticas e fazer as análises biológicas dos resultados, resultando em uma melhora no processo de avaliação de montagem feito por biólogos. A Seção 6.1 apresenta detalhes da infraestrutura e a Seção 6.2 os descreve detalhes do desenvolvimento.

6.1.
Infraestrutura

A aplicação utiliza a linguagem de programação C# usando *MONOdevelop*. *MONOdevelop* pode ser executado nos sistemas operativos Linux, BSD, UNIX, Mac OS X, Solaris e Windows.

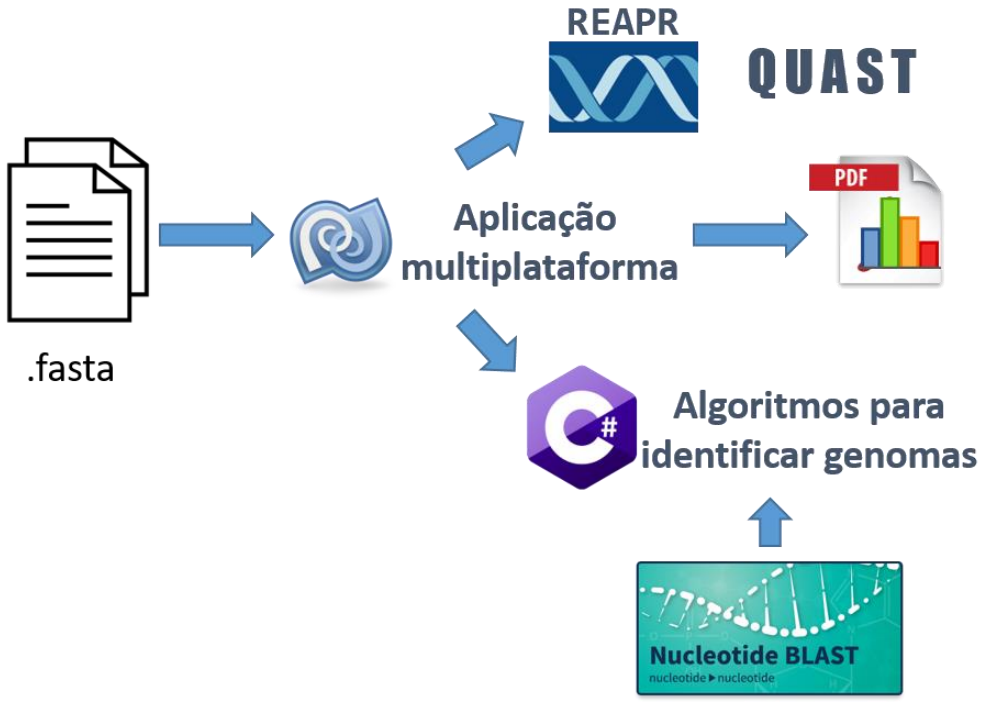


Figura 10. Arquitetura da Aplicação

Como mostrado na Figura 10, a aplicação recebe os arquivos .fasta dos genomas montados. A aplicação aciona os algoritmos para identificar genomas na montagem. Esses algoritmos usam o *blastn* para achar o alinhamento entre os genomas de referência e uma montagem. A aplicação também computa as estatísticas geradas pelas ferramentas *QUAST* e REAPR. Finalmente gera um relatório *PDF* com resultados comparativos entre as diferentes montagens recebidas que ajuda a determinar o melhor resultado.

6.2. Desenvolvimento

A continuação mostramos como a nossa ferramenta funciona. A Figura 11 apresenta a tela principal da ferramenta.

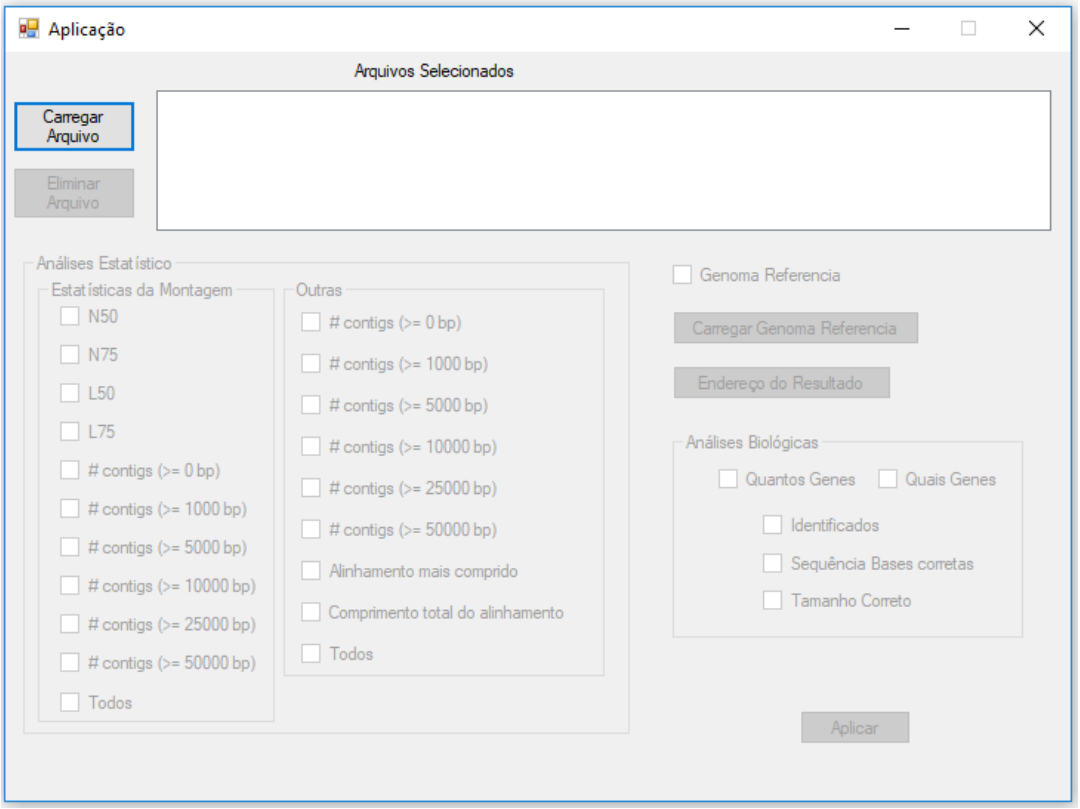


Figura 11. Tela da aplicação

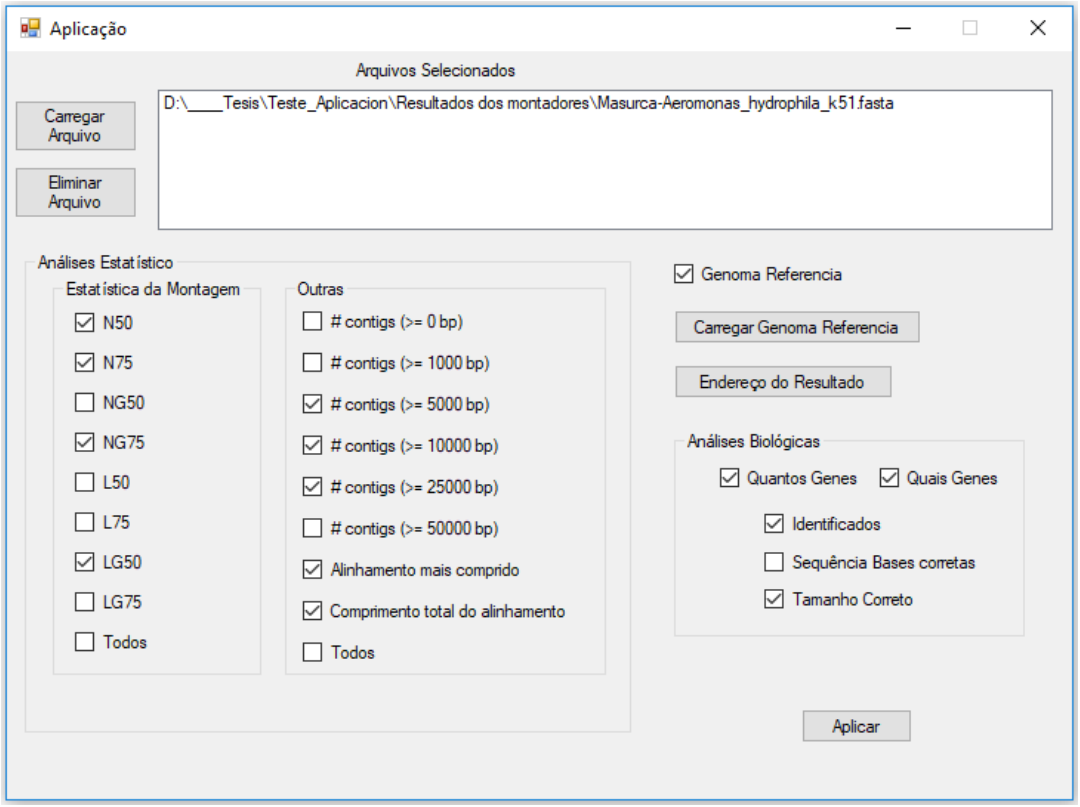


Figura 12. Tela da Aplicação escolhendo Genoma de Referência para as Análises

Primeiramente temos que carregar os genomas resultados de uma montagem, isto será feito mediante o botão *Carregar Arquivo*. Podem ser carregados resultados de diferentes montadores ou de um mesmo montador com diferentes valores de *k*. Depois de ter carregado(s) o(s) arquivo(s), têm-se várias opções dependendo de se tem ou não o genoma de referência, segundo mostra a Figura 12. Na Figura 13 podemos ver as opções disponíveis sem o genoma de referência. Neste caso, aparecem menos opções para analisar e carece das análises biológicas.

Ao ter já selecionados os parâmetros que desejamos computar, então pode dar click na opção *aplicar*. Nossa ferramenta fará o cálculo das estatísticas e a análise para identificar genes em cada uma das montagens para comparar (arquivo carregados). Finalmente um PDF é gerado com os resultados e com os gráficos comparativos das diferentes montagens. A aplicação também consegue executar comandos de Linux desde a linguagem C#.

A seguir uma breve explicação das principias tarefas da aplicação:

- Executar o blastn Figura 14 e computar o algoritmo para fazer as análises biológica de identificação de genes.
- Executar o *QUAST* Figura 15 e o REAR e obter as análises estatísticas.
- Exportação dos resultados em PDF.

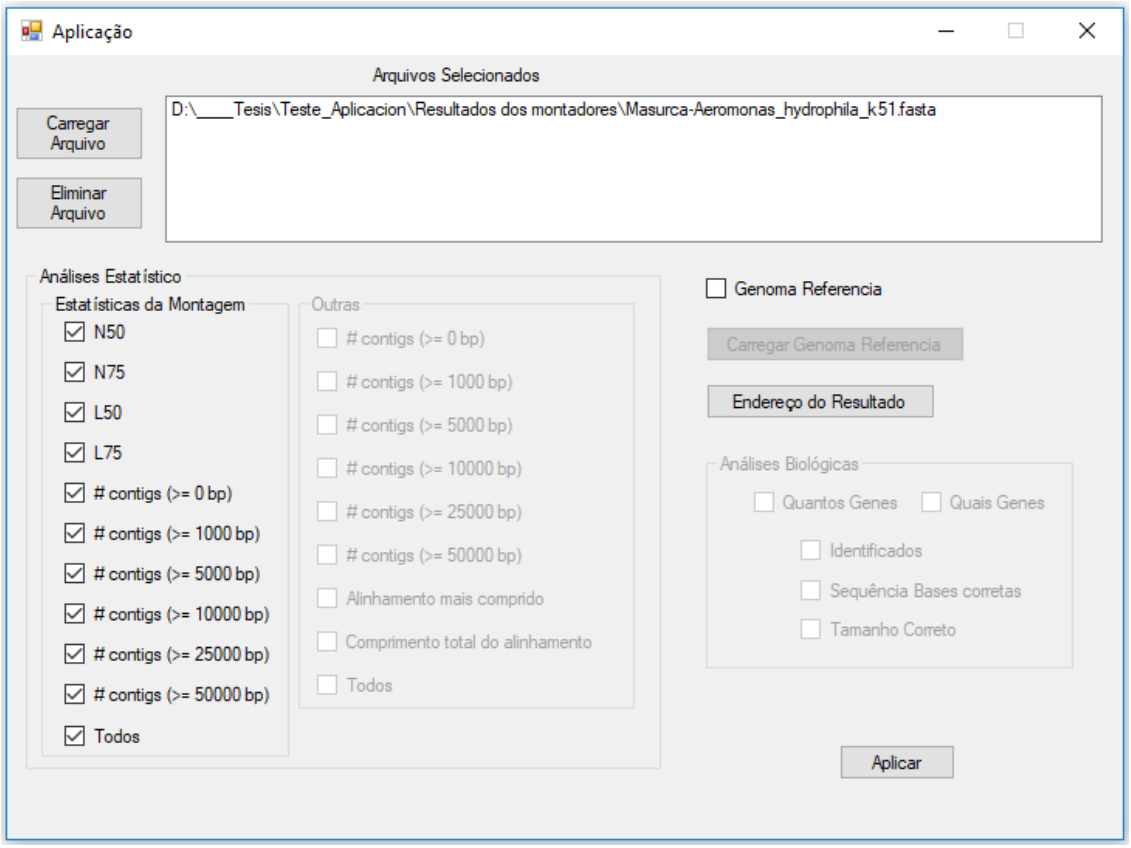


Figura 13. Tela da Aplicação sem Genoma de Referência para as Análises

```
ProcessStartInfo psi;
string[] nome_arq_temp;
string[] nome_arq;

for (int i = 0; i < endereco_arquivos.Count; i++)
{
    nome_arq_temp = endereco_arquivos[i].Split(Path.DirectorySeparatorChar);
    nome_arq = nome_arq_temp[nome_arq_temp.Length - 1].Split('.');

    psi = new ProcessStartInfo();
    psi.FileName = "/home/biobd/Alejandro_Tesis/blast/bin/blastn";
    psi.UseShellExecute = false;
    psi.Arguments = "-max_target_seqs 500 -num_threads 16 -outfmt " + @"""6 qseqid sseqid pident ppos score bitscore evalue qstart qend sstart
                    send qlen slen""" + " -query " + endereco_arquivos[i] + " -db " + endereco + Path.DirectorySeparatorChar + nome_arq[0]
                    + Path.DirectorySeparatorChar + nome_arq[0] + " -out " + endereco + Path.DirectorySeparatorChar + nome_arq[0] + ".blast";
    psi.RedirectStandardOutput = true;
    Process p = Process.Start(psi);
    Console.WriteLine(p.StandardOutput.ReadToEnd());
    p.WaitForExit();
    p.Close();
}
```

Figura 14. Código onde é executado o Blastn desde a linguagem C#

```
ProcessStartInfo psi;
string[] nome_arq_temp;
string[] nome_arq;

for (int i = 0; i < endereco_arquivos.Count; i++)
{
    nome_arq_temp = endereco_arquivos[i].Split(Path.DirectorySeparatorChar);
    nome_arq = nome_arq_temp[nome_arq_temp.Length - 1].Split('.');

    psi = new ProcessStartInfo();
    psi.FileName = "quast.py";
    psi.UseShellExecute = false;

    if (endereco_genoma_referencia == "")
        psi.Arguments = endereco_arquivos[i] + " -o " + endereco + Path.DirectorySeparatorChar + "output_" + nome_arq[0] + "_quast";
    else
        psi.Arguments = endereco_arquivos[i] + " -R " + endereco_genoma_referencia + " -o " + endereco + Path.DirectorySeparatorChar
            + "output_" + nome_arq[0] + "_quast";
    psi.RedirectStandardOutput = true;
    Process p = Process.Start(psi);
    Console.WriteLine(p.StandardOutput.ReadToEnd());
    p.WaitForExit();
    p.Close();
}
```

Figura 15. Código onde é executado o Quast desde a linguagem C#

7
Conclusão

Nesta dissertação foram analisados os diferentes problemas para obter a melhor montagem de um genoma. Para diminuir consideravelmente a complexidade das tarefas que um biólogo tem que executar para obter uma montagem e com uma boa qualidade foram realizados testes comparativos de uso da memória, o tempo de execução, estatísticas sobre a montagem e estudo dos genes presentes no resultado envolvendo cinco dos principais montadores de genoma conhecidos na literatura e oito genomas de bactérias. As estatísticas para avaliar a montagem foram obtidas usando as ferramentas *QUAST* e REAPR. Foi proposto e desenvolvido um algoritmo que faz a análise de quantos e quais genes do genoma de referência podem ser identificados em uma montagem; se os genes identificados têm o tamanho correto e se os genes identificados têm a sequência de bases correta. O algoritmo usa a ferramenta *blastn* para obter esses valores.

Como resultados de todos nossos testes e análises concluímos que o SPAdes é o melhor montador. Para isto, a abordagem escolhida foi associar um valor peso aos parâmetros e calcular a média ponderada. A associação dos pesos aos parâmetros foi feita seguindo um *ranking* de importância e priorizando os que estão relacionados com as análises biológicas.

Também confirmamos o que outras fontes da literatura já afirmaram "que o valor de *k* influência no resultado da montagem", mais que existe uma grande probabilidade que para valores de *k* que estão perto do comprimento do *read* são obtidos os piores resultados nas análises biológicas, já nas análises de eficiência e consumo de RAM o resultado é contrário.

Analisando a ferramenta *KmerGenie*, que faz recomendação de valores de *k* para realizar a montagem. Fizemos uma comparação com o montador SPAdes, que foi o melhor, em nos oito *dataset* e chegamos à conclusão que em muitos casos o valor de *k* recomendado por *KmerGenie* não traz consigo os melhores resultados, obtivemos valores de *k* com resultados melhores. Portanto, essa ferramenta não sempre recomenda o *k* com o qual se obtêm os melhores resultados.

Também apreciamos que na maioria dos casos quando o valor de N50 e número de *contigs* são melhores também é melhor a porcentagem de genes identificados com bases corretas e exatamente iguais. Além disso, podemos ver que em cada *dataset* os melhores *k* não são valores próximos, na verdade são bastante variados, pelo que não foi possível determinar qual poderiam ser os melhores *k*.

Por último, é proposta uma ferramenta que faz todas as análises assim como a avaliação do resultado em um modulo só. A ferramenta também cria relatórios com informações e gráficos comparativos que ajudam a escolher a melhor montagem.

8
Referências bibliográficas

(US), N. C. FOR B. I. **BLAST (r) Command Line Applications User Manual**.

ALTSCHULL, S. et al. Basic local alignment search tool. **J. mol. Biol**, 1990.

BAKER, M. De novo genome assembly: what every biologist should know. **Nature methods**, 2012.

BANKEVICH, A.; NURK, S.; ANTIPOV, D. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. **Journal of**, 2012a.

BANKEVICH, A.; NURK, S.; ANTIPOV, D. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. **Journal of**, 2012b.

CHAISSON, M.; PEVZNER, P.; TANG, H. Fragment assembly with short reads. **Bioinformatics**, 2004.

CHERUKURI, Y.; JANGA, S. Benchmarking of de novo assembly algorithms for Nanopore data reveals optimal performance of OLC approaches. **BMC genomics**, 2016.

CHIKHI, R.; MEDVEDEV, P. Informed and automated k-mer size selection for genome assembly. **Bioinformatics**, 2013.

CHIKHI, R.; RIZK, G. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. **International Workshop on Algorithms in Bioinformatics**, 2012.

COMPEAU, P.; PEVZNER, P.; TESLER, G. How to apply de Bruijn graphs to genome assembly. **Nature biotechnology**, 2011.

CONWAY, T.; BROMAGE, A. Succinct data structures for assembling large genomes. **Bioinformatics**, 2011.

DAVEY, J. et al. Genome-wide genetic marker discovery and genotyping using next-generation sequencing. **Nature Reviews**, 2011.

EARL, D. et al. Assemblathon 1: a competitive assessment of de novo short read assembly methods. **Genome**, 2011.

GUREVICH, A. et al. QUAST: quality assessment tool for genome assemblies. **Bioinformatics**, 2013.

HUNT, M.; KIKUCHI, T.; SANDERS, M. REAPR: a universal tool for genome assembly evaluation. **Genome**, 2013.

JÜNEMANN, S. et al. GABenchToB: A Genome Assembly Benchmark Tuned on

Bacteria and. 2014.

KINGSFORD, C.; SCHATZ, M.; POP, M. Assembly complexity of prokaryotic genomes using short reads. **BMC**, 2010.

LI, H.; HOMER, N. A survey of sequence alignment algorithms for next-generation sequencing. **Briefings in bioinformatics**, 2010.

LI, R. et al. De novo assembly of human genomes with massively parallel short read sequencing. **Genome**, 2010.

LUO, R. et al. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. 2012.

LUQUE, G.; ALBA, E. Metaheuristics for the DNA fragment assembly problem. **International Journal of Computational**, 2005.

MADDEN, T. The BLAST sequence analysis tool. 2013.

MAGOC, T. et al. GAGE-B: an evaluation of genome assemblers for bacterial organisms. 2013.

MILLER, J.; KOREN, S.; SUTTON, G. Assembly algorithms for next-generation sequencing data. **Genomics**, 2010.

NAGARAJAN, N.; POP, M. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. **Journal of computational biology**, 2009.

NAGARAJAN, N.; POP, M. Sequence assembly demystified. **Nature Reviews Genetics**, 2013.

PASZKIEWICZ, K.; STUDHOLME, D. De novo assembly of short sequence reads. **Briefings in bioinformatics**, 2010.

PÉREZ, N.; GUTIERREZ, M.; VERA, N. Computational performance assessment of k-mer counting algorithms. **Journal of Computational**, 2016.

POP, M. Genome assembly reborn: recent computational challenges. **Briefings in bioinformatics**, 2009.

RHOADS, A. et al. PacBio sequencing and its applications. **Elsevier**, [s.d.].

SALIKHOV, K.; SACOMOTO, G.; KUCHEROV, G. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. **International Workshop on**, 2013.

SALZBERG, S. et al. GAGE: A critical evaluation of genome assemblies and assembly algorithms. **Genome**, 2012.

SANGER, F. Nucleotide sequence of bacteriophage (D X174 DNA. 1977.

SCHATZ, M.; DELCHER, A.; SALZBERG, S. Assembly of large genomes using second-generation sequencing. **Genome research**, 2010.

SHENDURE, J.; BIOTECHNOLOGY, E. A.-N.; 2012, UNDEFINED. The

expanding scope of DNA sequencing. **nature.com**, [s.d.].

SHENDURE, J.; JI, H. Next-generation DNA sequencing. **Nature biotechnology**, 2008.

ZERBINO, D.; BIRNEY, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. **Genome research**, 2008.

ZHANG, W. et al. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. **PloS one**, 2011.

ZIMIN, A. et al. The MaSuRCA genome assembler. 2013.