UNIVERSIDADE FEDERAL DO PARÁ
NÚCLEO DE DESENVOLVIMENTO AMAZÔNICO EM ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

ANTONIO SÉRGIO CRUZ GAIA

FERRAMENTA BASEADA EM CUCKOO FILTER PARA REMOÇÃO DE REDUNDÂNCIA EM DADOS DE SEQUENCIADORES DE SEGUNDA GERAÇÃO (NGS - NEXT GENERATION SEQUENCING)

Tucuruí-PA

2019

ANTONIO SÉRGIO CRUZ GAIA

FERRAMENTA BASEADA EM CUCKOO FILTER PARA REMOÇÃO DE REDUNDÂNCIA EM DADOS DE SEQUENCIADORES DE SEGUNDA GERAÇÃO (NGS - NEXT GENERATION SEQUENCING)

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Profº Adonney Allan de O. Veras

Tucuruí-PA

2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

G137f Gaia, Antonio Sérgio Cruz
Ferramenta Baseada em Cuckoo Filter para Remoção de Redundância em Dados de Sequenciadores de Segunda Geração (NGS - Next Generation Sequencing) / Antonio Sérgio Cruz Gaia. — 2019.
114 f. : il. color.

Orientador(a): Prof. Dr. Adonney Allan de Oliveira Veras
Dissertação (Mestrado) - Programa de Pós-Graduação em Computação Aplicada, Núcleo de Desenvolvimento Amazônico em Engenharia, Universidade Federal do Pará, Tucuruí, 2019.

1. NGS. 2. Redundância. 3. Bioinformática. 4. Software. I. Título.

CDD 005

ANTONIO SÉRGIO CRUZ GAIA

FERRAMENTA BASEADA EM CUCKOO FILTER PARA REMOÇÃO DE REDUNDÂNCIA EM DADOS DE SEQUENCIADORES DE SEGUNDA GERAÇÃO (NGS - NEXT GENERATION SEQUENCING)

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Profº Adonney Allan de O. Veras

Aprovada em 02 de dezembro de 2019.

BANCA EXAMINADORA:

Prof. Dr. Adonney Allan de Oliveira Veras (Presidente)

Prof. Dr. Bruno Merlin - UFPA (Membro Interno)

Profa. Dra. Jorianne Thyeska Castro Alves (Membro Externa)

Prof. Dr. Pablo Henrique Caracciolo Gomes de Sá (Membro Externo)

Dedico este trabalho às minhas filhas
Laura e Maria Lia.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado saúde e força para superar as dificuldades.

Agradeço a minha esposa Karla Serrão e as nossas filhas Laura e Maria Lia por todo amor, carinho e apoio nos momentos mais difíceis. Vocês são a Luz da minha vida.

Agradeço aos meus pais Luzinan e Maria Isabel pelo exemplo de dignidade e perseverança, pelo amor e apoio a mim concedidos e por me ensinarem a nunca desistir, meus eternos agradecimentos.

Agradeço aos meus irmãos pela amizade, carinho e por sempre torcerem por mim.

Agradeço ao meu orientador Prof. Allan Veras por compartilhar seu conhecimento e por estar sempre disponível a esclarecer minhas dúvidas. Muito obrigado por sua contribuição, paciência e oportunidade de realização deste trabalho.

Agradeço a todos os professores e colaboradores do Programa de Pós-graduação em Computação Aplicada pela colaboração e paciência.

Agradeço aos colegas da turma PPCA 2017, com os quais dividi umas das etapas mais importante da minha vida.

Agradeço aos alunos de Engenharia da Computação Eurialdo e Vivian e a aluna de mestrado Mônica Oliveira pela contribuição e ajuda no desenvolvimento deste trabalho, meu muito obrigado.

Finalmente, agradeço a todos que aqui não foram citados, mas que de alguma forma contribuiram na realização deste trabalho.

RESUMO

As plataformas de sequenciamento de segunda geração também conhecidas como NGS – *Next Generation Sequencing* produzem grande volume de dados, o que demanda alta complexidade e custo computacional no processamento destes dados. Essas plataformas geram leituras duplicadas que surgem na preparação da biblioteca genômica e são introduzidas na etapa de amplificação por PCR (*Polymerase Chain Reaction*). Essa redundância de leituras pode aumentar os requisitos computacionais e tempo de processamento de análises subsequentes (por exemplo, a montagem *de novo)*. Para reduzir o custo computacional dessas análises é necessário realizar a remoção dessas leituras do conjunto de dados do organismo sequenciado. Neste trabalho apresentamos o NGSReadsTreatment uma ferramenta computacional para a remoção de leituras duplicadas em conjuntos de dados pareados ou fragmentos. A entrada de dados para o NGSReadsTreatment consiste em leituras oriundas de qualquer plataforma de sequenciamento com o mesmo ou diferentes comprimentos. A sua *engine* utiliza a estrutura probabilística *Cuckoo Filter* para identificar e remover as leituras redundantes, a identificação é feita comparando as leituras entre si, assim, nenhum pré-requisito é necessário além do conjunto de leituras. A validação da ferramenta foi realizada utilizando-se conjuntos de dados reais e simulados. Para aferir a eficiência da ferramenta, a mesma foi comparada com outras ferramentas de remoção de redundância. Os resultados indicam a eficiência do NGSReadsTreatment, pois obteve-se melhor resultado, tanto na quantidade de redundâncias removidas quanto no uso de memória em todos os testes realizados. Desenvolvido em JAVA, o NGSReadsTreatment é compatível com os sistemas operacionais UNIX/Linux e Windows e dispoẽ de uma versão com interface gráfica para facilitar seu uso.

O NGSReadsTreatment e o guia do usuário estão disponíveis para download em https://sourceforge.net/projects/ngsreadstreatment/.

Palavras-chave: NGS. Redundância. Bioinformática. Software.

ABSTRACT

The second-generation sequencing platforms, also known as NGS – Next Generation Sequencing, produce a great amount of data, which demands high complexity and computational cost in the processing of these data. These platforms generate duplicated reads that come from the preparation of the genomic library and are included in the amplification stage by PCR (Polymerase Chain Reaction). This redundancy can increase the computational requirements and processing time of subsequent analyses (for instance, *de novo* assembly). To reduce the computational cost of theses analyses, it is necessary to remove these reads from the data set of the sequenced organism. In this work, we present the NGSReadsTreatment, a computational tool to remove duplicated reads in paired-end or single-end data sets. The input for NGSReadsTreatment consists of reads from any sequencing platform with same or different read lengths. Its engine uses a Cuckoo Filter probabilistic structure to identify and remove redundant readings. The identification is done by comparing the reads among themselves, this way, not any pre-requisite is necessary besides the reads set. The validation of the tool was carried out by using a set of real and simulated data. To assess the efficiency of the tool, it was compared to other tools of redundancy removal. The results indicate the efficiency of the NGSReadsTreatment, for it produced the best outcome, both in the number of redundancies removed and the use of memory, in all tests done. Developed in JAVA, the NGSReadsTreatment is compatible with UNIX/Linux and Windows operating systems and has a version with a graphic interface to facilitate its use.

The NGSReadsTreatment and user guide are available for download at https://sourceforge.net/projects/ngsreadstreatment/.

Keywords: NGS. Redundancy. Bioinformatics. Software.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ASCII	Código Padrão Americano para o Intercâmbio de Informação.
CPU	*Central processing unit*. Unidade Central de Processamento.
DER	Diagrama Entidade Relacionamento.
DNA	*Deoxyribonucleic acid.* Ácido Desoxirribonucleico.
Gb	Gigabases.
GB	Gigabyte.
GPU	*Graphic Processing Unit.* Unidade de Processamento Gráfico.
HD	*Hard Disk*. Disco Rígido.
HDFS	*Hadoop Distributed File System.*
JAR	*Java Archive.*
JDK	*Java Development Kit.*
JRE	*Java Runtime Enviroment.*
MB	*Megabyte.*
MPI	*Message Passing Interface.*
NCBI	*National Center for Biotechnology Information.*
NGS	*Next Generation Sequencing.* Sequenciadores de nova geração.
Pb	Pares de base.
PCR	*Polymerase Chain Reaction.* Reação em Cadeia da Polimerase.
SBL	Sequenciamento por Ligação.
SBS	Sequenciamento por Síntese.
SGBD	Sistema Gerenciador de Banco de Dados.
SNP	*Single nucleotide polymorphism.* Polimorfismo de Nucleotídeo Único.
SRA	*Sequence Read Archive*.

SUMÁRIO

1 INTRODUÇÃO

A genetica é a especialidade da biologia que estuda o material genético (DNA e RNA), incluindo genes, questões de hereditariedade e a forma como características são transmitidas de geração a geração. Esta ciência se divide em genética clássica, molecular e de populações. A genética clássica abrange os princípios básicos da hereditariedade e como as características são transmitidas de uma geração para outra. A genética molecular estuda a estrutra e função dos genes a nível molecular: como as informações genéticas são codificadas, replicadas e expressas. Na genética de populações os genes são estudados por avaliação da variabilidade entre indivíduos de uma população (PIERCE, 2016; SNUSTAD & SIMMONS, 2017). Este trabalho tem o foco em genética molecular por tratar especificamente de leituras produzidas após o processo de sequenciamento de amostras de DNA de um organismo.

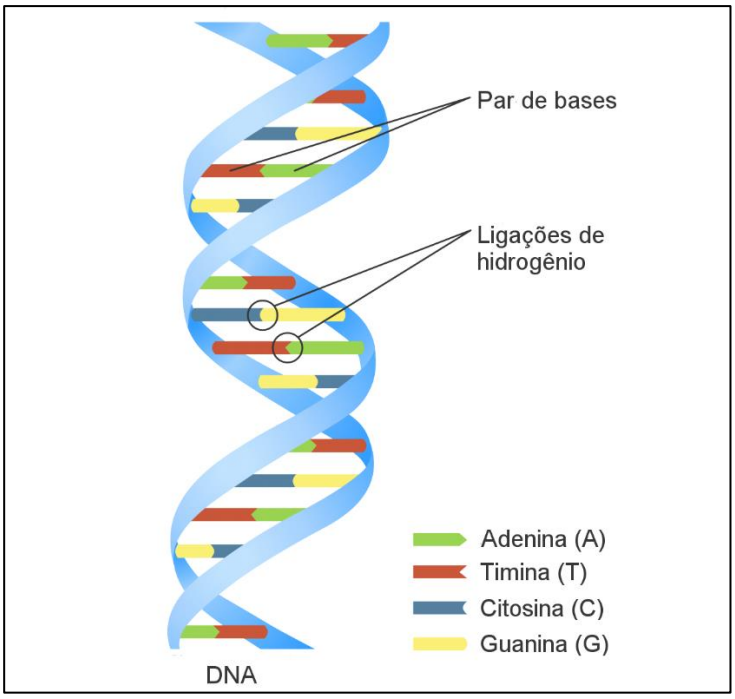
O DNA contém toda a informação genética dos seres vivos e tem a função de armazenar e transmitir estas informações a todos os organismos (SNUSTAD & SIMMONS, 2017). As pesquisas realizadas por esta ciência proporcionam grande impacto sobre os seres humanos, como por exemplo, aplicações da genética na medicina permitem a detecção de doenças, diagnósticos clínicos e, mais recentemente, o desenvolvimento da medicina personalizada (DIJK *et al*., 2014).

Na agricultura é aplicada nas principais culturas e animais através de alterações genéticas que aumentam o rendimento, resistência a doenças e pragas, além do aumento das qualidades nutricionais dos alimentos produzidos (PIERCE, 2016). Também através da genética é possível conhecermos a evolução dos organismos por meio do estudo da transmissão das características hereditárias ao longo de suas gerações, além de auxiliar no entendimento dos mais diversos organismos presentes no planeta. (SNUSTAD & SIMMONS, 2017).

Em 1953, o estudo de James Watson e Francis Crick descobriu que o DNA possui uma estrutura molecular em forma de dupla-hélice composta por dois filamentos de DNA enrolados lado a lado de modo espiral (Figura 1). Esses filamentos com 4 diferentes bases químicas no DNA armazenam a informação genética de modo muito parecido ao modo como as séries de zeros (0) e uns (1)

codificam informação na computação (GRIFFITHS *et al*., 2016). Essas 4 bases: adenina (A), timina (T), guanina (G) e citosina (C) são ligadas à sua base complementar (par de bases) através de ligações de hidrogênios e, em um filamento a adenina sempre estará pareada com a timina e a guanina sempre estará pareada com a citosina. A sequência dessas bases (A, T, G, C) representa a informação genética codificada pela molécula de DNA.

Figura 1 – Representação da estrutura molecular do DNA



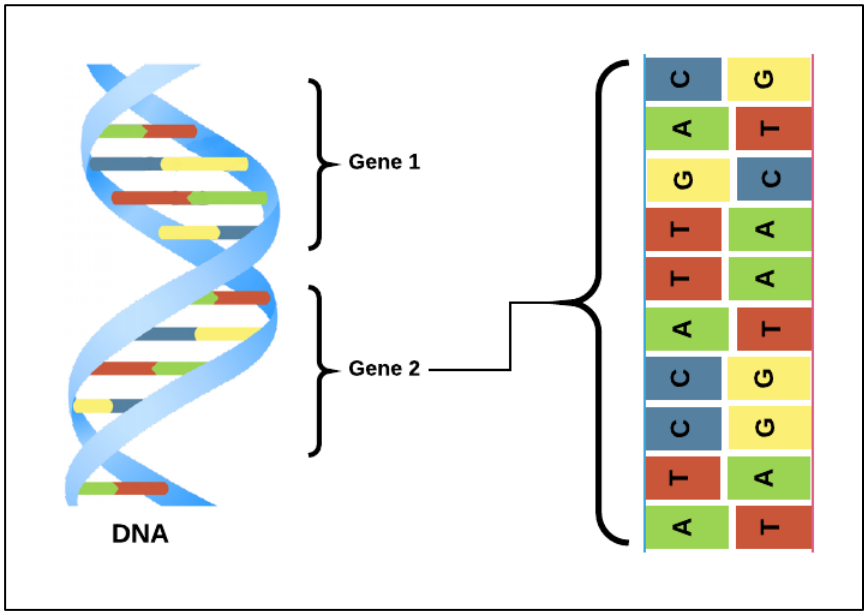
Fonte original Genome Research Limited. Adaptado e traduzido de The Human Origin Project[1]

O processo utilizado para identificar a sequência de bases A, T, G, C em uma molécula de DNA é denominado de sequenciamento de DNA (GRIFFITHS *et al*., 2016). Uma sequência específica de bases (A, T, G, C) ao longo de uma molécula de DNA compõe um gene (Figura 2). Logo, os genes são segmentos de DNA com uma sequência de bases própria que codificam uma determinada informação genética, sendo que cada molécula pode conter milhares de diferentes genes. O conjunto de todos os diferentes genes de um organismo é denominado genoma.

[1] https://humanoriginproject.com/dna-structure-function/

Sequenciar um genoma significa sequenciar toda a sequencia de nucleotídeos do organismo (SNUSTAD & SIMMONS, 2017).

Figura 2 – Representação de um gene em uma molécula de DNA



Fonte: Adaptado de NetNature[2]

Inicialmente, a genética concentrava-se no estudo de genes isolados, porém, atualmente com o uso dos sequenciadores de DNA é possível realizar o estudo do conteúdo gênico completo do organismo alvo. O avanço no processo de sequenciamento aliado ao desenvolvimento de ferramentas de biotecnologia e bioinformática tem permitido o estudo de genomas de organismo cada vez mais complexos (REUTER *et al*., 2015).

Sequenciadores são equipamentos que promovem a identificação de bases nitrogenadas que compõem o DNA (bases nitrogenadas A, T, G e C). Através do sequenciamento é possível fazer diversas análises como: Genômica[3], Transcriptômica[4], Metagenômica[5] e etc. Alguns exemplos destes equipamentos

[2] https://netnature.wordpress.com/2015/05/08/a-origem-de-novos-genes-e-pseudogenes/
[3] É um ramo da genética que estuda o genoma completo ou parcial de um organismo.
[4] Refere-se ao estudo das moléculas de RNA formadas no processo de transcrição (DNA → RNA mensageiro).
[5] É o estudo de genomas completos ou fragmentos de DNA de microrganismos presente em um determinado ambiente.

são: Applied Biosystems 3730xl, Roche 454, Illumina MiSeq, SOLiD, Ion Proton, Oxford Nanopore, etc. Eles são categorizados em três gerações: primeira, segunda e terceira geração.

1.1 Sequenciadores de Primeira Geração

Os sequenciadores de primeira geração são caracterizados por utilizar o método de sequenciamento proposto por Frederick Sanger em meados de 1970. Estes equipamentos possuíam uma alta eficiência e já não utilizavam isótopos radioativos. Este método era considerado manual e nas décadas seguintes sofreu evoluções tornando-se um processo automatizado na década de 90 (HEATHER *et al*., 2016). Alguns exemplos destes sequenciadores são: Applied Biosystems 3730xl e Applied Biosystems SeqStudio.

Apesar da evolução, estas plataformas de sequenciamento tem um elevado custo por base sequenciada e um baixo rendimento. Isso dificultava o uso desses sequenciadores em larga escala. Para tentar sanar esses problemas, em meados de 2005 foram lançados no mercado os sequenciadores de segunda geração (GUZVIC, 2013).

1.2 Sequenciadores de Segunda Geração

Em 2005, a ROCHE disponibilizou para o mercado a primeira plataforma de sequenciamento de segunda geração, tornando-se popularmente conhecida como sequenciadores da próxima geração (NGS - Next Generation Sequencing). Estes sequenciadores reduziram drasticamente o custo por base sequenciada, quando comparado com os de primeira geração. Além do custo, o tempo de sequenciamento também foi reduzido e isso possibilitou o uso dessas plataformas em larga escala, o que impulsionou os projetos de estudos genômicos (HEATHER *et al*., 2016).

Devido à impossibilidade de sequenciamento da molécula de DNA de uma só vez por este processo, é necessária a quebra desta molécula em vários fragmentos,

o que pode ser feito por meio enzimático ou por sonicação. Contudo, durante a preparação da biblioteca genômica utilizada por estes sequenciadores, a amostra do fragmento de DNA sofre um processo de clonagem e isso resulta em uma super-representação deste fragmento fazendo com que o organismo a ser sequenciado apresente seu tamanho multiplicado por diversas vezes (esta super-representação é denominada de cobertura). Essa amplificação clonal faz com que o volume de dados produzido por estes sequenciadores seja elevado, aumentando a complexidade computacional necessária para processar estes dados (REUTER *et al*., 2015). Podemos citar como exemplo destes sequenciadores: Roche 454, Illumina MiSeq Series, SOLiD, Ion Proton, etc.

1.3 Sequenciadores de Terceira Geração

Estes sequenciadores ao contrário dos de segunda geração não fazem uso da amplificação clonal, isto faz com que o volume de dados gerado seja menor. Também possuem um baixo custo e são bastante rápidos para realizar o sequenciamento. Porém, estas plataformas ainda não são utilizadas em larga escala por possuírem uma alta taxa de erros associados à base sequenciada, quando comparadas com a dos sequenciadores de segunda geração, fato que faz com que os sequenciadores de segunda geração sejam ainda amplamente utilizados em pesquisas genéticas (GUZVIC, 2013). Alguns exemplos destas plataformas são: Pacbio e Oxford Nanopore.

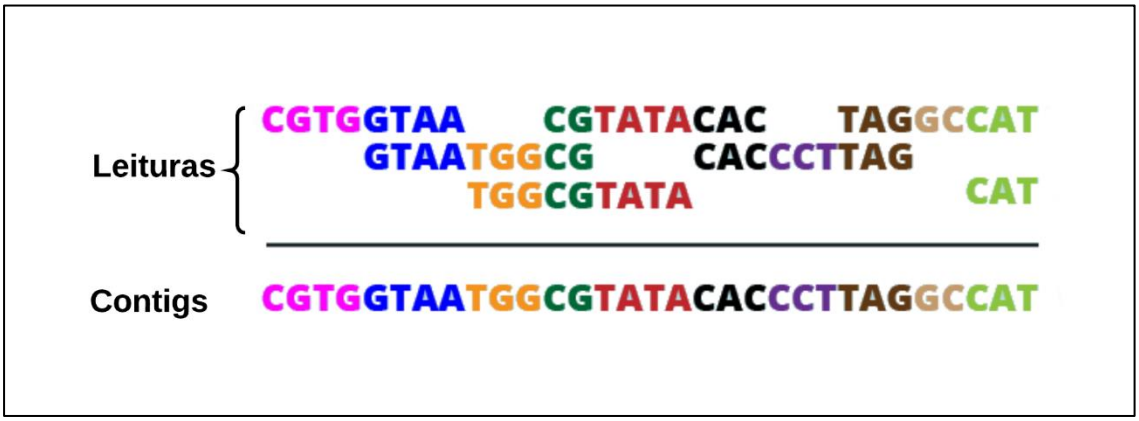
1.4 Justificativa

Devido à elevada taxa de erros das plataformas de terceira geração o escopo deste trabalho abrange somente sequenciadores de segunda geração. Estas plataformas são as mais utilizadas atualmente por possuírem um baixo custo e menor tempo de sequenciamento além de uma baixa taxa de erros associado à base sequenciada (HENSON *et al*., 2012).

Antes de realizar o sequenciamento é necessário preparar a biblioteca genômica e umas das etapas de preparação é a amplificação do fragmento de DNA efetuada pela reação em cadeia da polimerase (PCR) (REUTER *et al*., 2015). A PCR super-representa o fragmento da amostra de DNA, com isso o organismo sequenciado apresenta seu tamanho multiplicado por várias vezes. Esta super-representação é importante para análises que necessitam de grande quantidade de uma sequência de DNA específica como: curadoria de *frameshift*[6], análise de polimorfismos de nucleotídeo único (SNP)[7], dentre outras (REUTER *et al*., 2015).

No entanto, algumas análises são afetadas por esta super-representação das leituras como, por exemplo, a montagem *de novo* (Figura 3), que consiste na reconstrução da sequencia original do genoma sem a utilização de uma sequência conhecida como referência. É realizada através da identificação da sopreposição de duas ou mais leituras, seguida da construção de sequências consenso. Essas sequências consenso são também chamadas de *contigs*[8] (HENSON *et al*., 2012).

Figura 3 - Figura ilustrando a montagem *de novo*



Fonte: Elaborada pelo autor

Buscando reduzir o impacto da super-representação das leituras em análises como a montagem *de novo*, vários autores indicam um tratamento prévio do

[6] Processo de correção de inserções, deleções e substituições inseridas na etapa de montagem da sequencia de DNA (AGUIAR, 2015).
[7] É uma variação na sequencia de DNA que ocorre quando um único nucleotídeo (A, T, G, C) no genoma difere entre indivíduos de uma espécie ou entre pares de cromossomos de um indivíduo.
[8] Sequências contínuas geradas com a sobreposição de duas ou mais leituras.

conjunto dos dados NGS para reduzir seu tamanho ou melhorar a qualidade dos dados (EL-METWALLY *et al.,* 2013). Umas das etapas de pré-processamento é a remoção de leituras duplicadas (ZHOU e ROKAS, 2014). Essas leituras duplicadas elevam os custos computacionais necessários para processamento dos dados, além de implicar no surgimento de falso-positivos com a sobreposição de *contigs* (EBBERT *et al*., 2016).

Diante disso, surgiram inúmeras ferramentas que utilizam diversas técnicas computacionais para remoção de leituras duplicadas produzidas por sequenciadores NGS. Dentre estas ferramentas podemos citar: O GPU-DupRemoval que utiliza processamento de unidades gráficas (GPU) para remover a redundância de leituras geradas pelas plataformas Illumina (MANCONI *et al*., 2016). Há também o MarDRe, que é uma ferramenta baseada no modelo de programação *MapReduce* para remoção de redundância de leituras fragmentos e pareadas em dados NGS (EXPÓSITO *et al*., 2017). Outros exemplos de ferramentas incluem ParDRe (GONZÁLEZ-DOMÍNGUEZ *et al*., 2016), Fulcrum (BURRIESCI *et al*., 2012), FastUniq (XU *et al*., 2012), CD-HIT (LI *et al*., 2006), Fastx-Toolkit Collapser (http://hannonlab.cshl.edu/fastx_toolkit).

Contudo, observa-se que essas ferramentas foram desenvolvidas para atender uma plataforma específica, são executadas em linhas de comandos extensas e complexas e estão disponíveis somente para sistemas operacionais Linux, sendo que a maioria requer um alto poder computacional para processamento dos dados sequenciados. Isso dificulta o uso destas ferramentas por pesquisadores e grupos de pesquisa que dispõem de pouco recurso computacional, o que impacta diretamente na produção científica destes pesquisadores (GOECKS *et al*., 2010).

Diante disso, este trabalho busca desenvolver uma ferramenta computacional para remoção de leituras duplicadas, com suporte para conjunto de dados em fragmentos ou pareados e independente de plataforma de sequenciamento.

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral desta pesquisa é desenvolver uma ferramenta computacional para remoção de leituras duplicadas, com suporte a dados em fragmentos ou pareados e independente de plataforma de sequenciamento.

1.5.2 Objetivos Específicos

Os objetivos específicos visam:

- Realizar testes com estruturas de dados otimizados para definição da melhor solução;
- Implementar algoritmo para remover redundância no arquivo de entrada;
- Desenvolver módulo de interface gráfica que permita: gerenciar projetos dos usuários, acompanhar o processamento e retomar o processo em caso de falhas;
- Modelar a base de dados de projetos;
- Realizar a validação da ferramenta.

1.6 Organização do trabalho

Os próximos capítulos estão organizados da seguinte forma:

No capítulo 2 são descritos os conceitos básicos dos (i) Sequenciadores de Segunda Geração; (ii) Tipos de Leituras dos Sequenciadores NGS; (iii) Arquivo FASTQ; (iv) *Strings* e redundância de *strings* e (v) Remoção de redundância em dados NGS.

No capítulo 3 são apresentados os trabalhos relacionados e considerados como mais relevantes a esta pesquisa.

No capítulo 4 é descrita a ferramenta proposta, apresentando: (i) a arquitetura utilizada, (ii) funcionalidades, (iii) diagrama de casos de usos e (iv) diagrama entidade relacionamento.

No capítulo 5 são descritos os materiais e métodos utilizados na pesquisa.

No capítulo 6 são apresentados os resultados e a discussão em relação aos resultados obtidos.

E por fim, no capítulo 7 são expostas as considerações finais desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

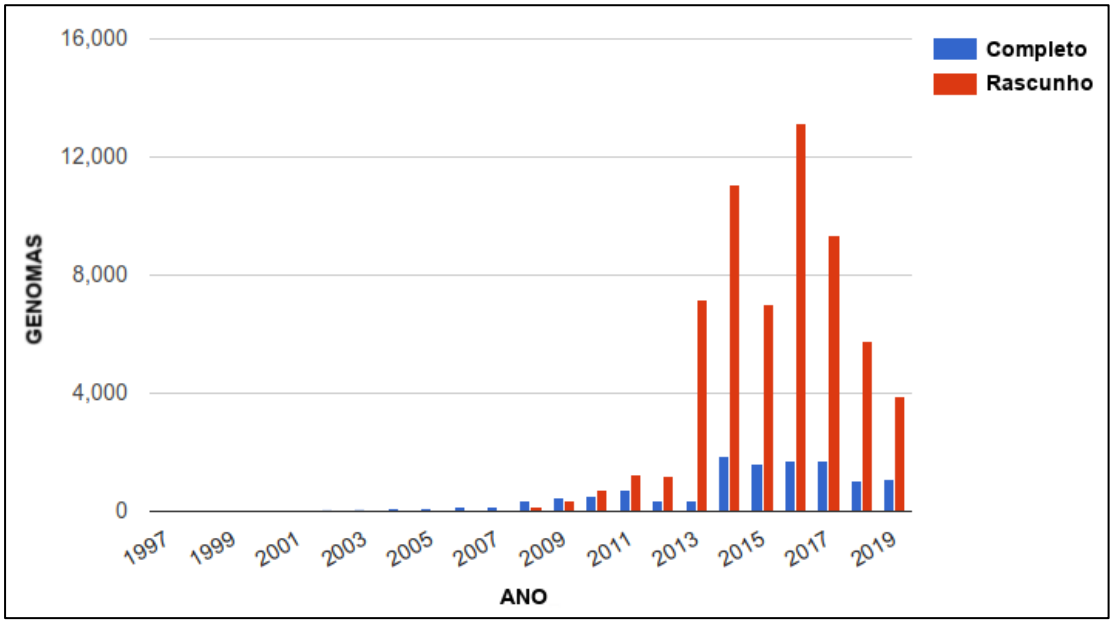
Neste capítulo serão abordados os sequenciadores de segunda geração que fazem parte do escopo deste trabalho, serão descritas as principais plataformas de sequenciamento NGS, os tipos de leituras produzidas por esses sequenciadores, o formato de arquivo padrão gerado por estes sequenciadores com as bases sequenciadas, o que são *strings* e redundância de *strings* e como é feita a remoção de redundância em dados NGS, abordando em detalhes a estrutura probabilística *Cuckoo Filter* (Fan *et al.*, 2014) que será utilizada para remoção de redundância das leituras produzidas por estas plataformas.

2.1 Sequenciadores de Segunda Geração

As tecnologias de sequenciamento de segunda geração revolucionaram as pesquisas biológicas em áreas como a genômica, genética, medicina e biotecnologia a partir de seu surgimento em meados de 2005. Dentre as suas principais características pode-se citar: a redução do custo e tempo de sequenciamento de genomas e a geração de um grande volume de dados, quando comparadas com o método de Sanger (HENSON *et al*., 2012).

A redução do custo e tempo de sequenciamento permitiu a utilização dessas plataformas em larga escala, impulsionando os estudos genômicos. Abaixo na Figura 4 podemos observar o crescimento desses estudos proporcionado pela utilização destas plataformas, tornando estas metodologias uma importante ferramenta para estudos em diversas áreas com aplicações que vão desde a detecção e tratamento de doenças até o aumento da produção de alimentos.

Figura 4 – Avanço dos estudos genômicos

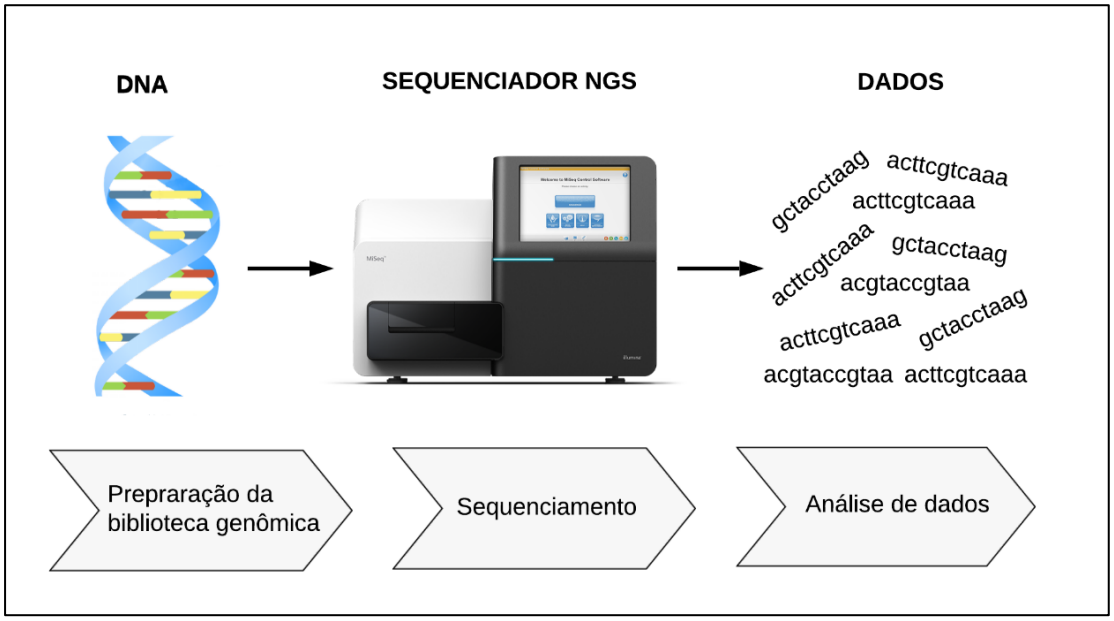


Fonte: Genome Online Database[9]

Essas plataformas se enquadram em duas categorias de abordagem de sequenciamento: Sequenciamento por Ligação (SBL) e Sequenciamento por Síntese (SBS). Nessas abordagens as reações de sequenciamento ocorrem por ligação de DNA (SBL) ou por ciclos de adição de bases nitrogenadas (SBS), sendo em seguida visualizados por processos de fluorescência, quimioluminescência (pirosequenciamento), ou alterações na corrente elétrica no chip de silício (GOODWIN *et al*., 2016). A Figura 5 ilustra o *workflow* básico desses sequenciadores NGS.

[9] https://gold.jgi.doe.gov/statistics

Figura 5 – *Workflow* básico dos sequenciadores NGS



Fonte: Adaptado e traduzido de ILVO[10]

2.1.1 Roche 454

A ROCHE 454 foi à primeira plataforma de sequenciamento NGS disponibilizada para o mercado em meados de 2005, e se baseava na tecnologia de pirosequenciamento. A primeira versão denominada 454 GS20 produzia leituras relativamente curtas (~150pb) quando comparadas com as produzidas nas plataformas de primeira geração (HENSON *et al*., 2012). A tecnologia de sequenciamento passou por evolução e o comprimento médio de leitura aumentou para ~700pb, porém essa evolução não foi acompanhada por uma melhoria na qualidade das leituras. (GILLES *et al*., 2011).

As principais vantagens dessas plataformas são: o tamanho de leituras produzidas (~700pb a ~1000pb), o alto rendimento – *throughput* e a disponibilidade de trabalhar com bibliotecas de fragmentos e pareadas. As principais desvantagens são a elevada taxa de erros em regiões homopoliméricas (que são sequências com repetições de bases nitrogenadas de único nucleotídeo, ex: AAAAAA, TTTTTT) e o

[10] https://www.ilvo.vlaanderen.be

alto custo de reagentes utilizados no sequenciamento (GUZVIC, 2013). A Figura 6 abaixo ilustra um modelo dessa plataforma.

Figura 6 – Sequenciador Roche 454 GS Junior

Fonte: Roche[11]

2.1.2 Illumina Genome Analyzer

Lançada em 2007 esta plataforma revolucionou o sequenciamento de genomas, sendo capaz de produzir um maior volume de dados quando comparado com a 454 da ROCHE (GUZVIC, 2013). Utiliza o método de sequenciamento por síntese (SBS) proposto por Sanger (SANGER *et al*., 1977). Ao longo do surgimento de novas versões, o comprimento de leituras e o *throughput* (rendimento) sofreram mudanças, passando de 35pb e 1Gb para 100pb e 600Gb na versão HiSeq 2000 (HENSON *et al*., 2012).

Estas plataformas têm alta precisão (taxa de erros <1%) e custos relativamente baixos, por isso são amplamente utilizadas em aplicações de montagem *de novo* de genomas e outros estudos (HENSON *et al*., 2012). Esses sequenciadores possuem como o erro mais comum a substituição de bases (DOHM *et al*., 2008). A Figura 7 exibe um modelo desse sequenciador.

[11] https://www.roche.com/media/releases/med-cor-2012-01-17t.htm

Figura 7 – Sequenciador Illumina MiSeq Series

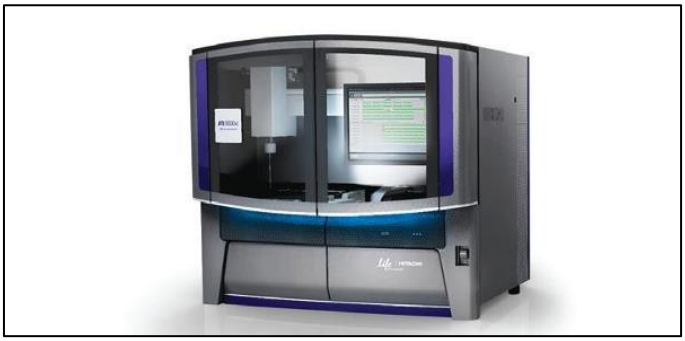
Fonte: Illumina[12]

2.1.3 SOLiD

A plataforma SOLiD se baseia no método de sequenciamento por ligação (SBL). Trabalha com leituras extremamente curtas, com tamanho que varia de ~30pb a ~50pb, o que acaba gerando um grande volume de dados. (HENSON *et al*., 2012).

As principais vantagens desses sequenciadores são a alta cobertura e acurácia das leituras. As leituras produzidas por esses sequenciadores são problemáticas para montagem *de novo* devido ao tamanho curto e também o grande volume de dados gerado, o que exige maior capacidade e poder computacional para realizar esse tipo de análise (GUZVIC, 2013). A Figura 8 demonstra um modelo dessa plataforma.

Figura 8 – Sequenciador SOLiD 5500xl



Fonte: ThermoFisher [13]

[12] https://www.illumina.com/systems/sequencing-platforms/miseq.html
[13] https://www.thermofisher.com/order/catalog/product/4460730#/4460730

2.2 Tipos de Leituras dos Sequenciadores NGS

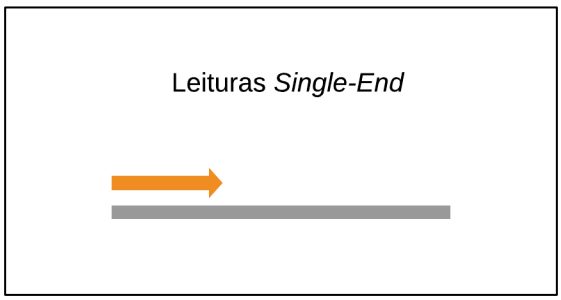
No processo de sequenciamento, as plataformas NGS utilizam bibliotecas genômicas que são compostas por fragmentos de DNA de um único organismo. Existem diversos protocolos para preparação da biblioteca genômica, porém todos têm em comum o fato de que fragmentos da molécula de DNA são ligados com adaptadores específicos de cada sequenciador. Em seguida, geralmente é executada a etapa de seleção de tamanho e eliminação de adaptadores livres. Por fim, é realizada a PCR para selecionar fragmentos que contém adaptadores nas duas extremidades e gerar quantidades suficientes (amplificação de fragmentos) para o sequenciamento (DIJK *et al*., 2014).

Nos sequenciadores NGS as leituras são produzidas de acordo com o tipo de biblioteca genômica e estas são classificadas como: fragmentos (*single-end*) e pareadas (*paired-end* e *mate-pair*):

2.2.1 Fragmentos (*Single-End*)

Nesse tipo de biblioteca os sequenciadores executam a leitura em uma única extremidade do fragmento. Como saída deste tipo de execução, tem-se um único arquivo com as bases sequenciadas (METZKER, 2010). A Figura 9 ilustra esse tipo de leitura.

Figura 9 - Ilustração de leituras *Single-End*



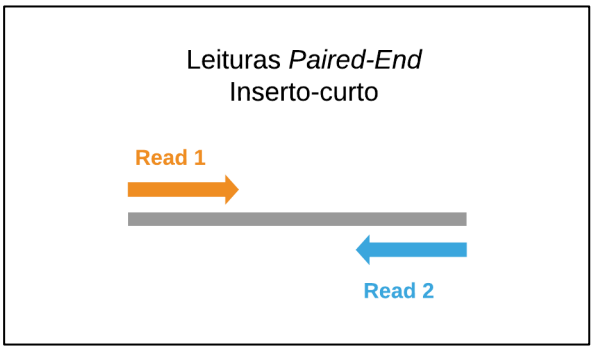
Fonte: Adaptado e traduzido de Illumina[14]

[14] https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html

2.2.2 Pareadas (*Paired-End*)

O sequenciamento deste tipo de biblioteca é executado nas duas extremidades do fragmento do DNA, para isso é preciso ligar adaptadores dos dois lados da sequência para sequenciá-los. Como resultado, são gerados dois arquivos pareados que contêm o sequenciamento do fragmento (inserto) de DNA (METZKER, 2010). Abaixo a Figura 10 ilustra o tipo de leitura *paired-end*.

Figura 10 - Ilustração de leituras *Paired-End*



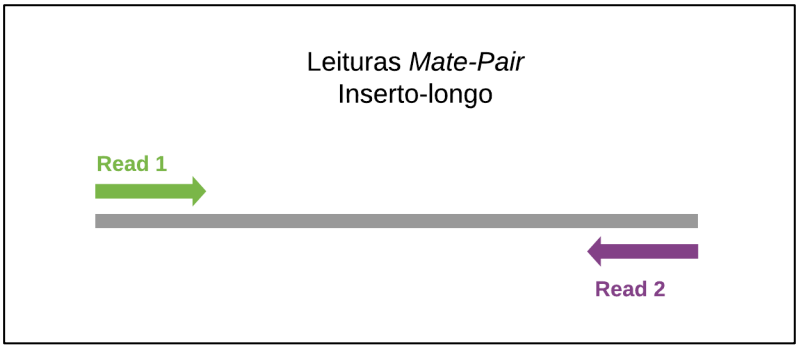
Fonte: Adaptado e traduzido de Illumina[15]

2.2.3 Pareadas (*Mate-Pair*)

O sequenciamento da biblioteca *mate-pair* (Figura 11) é similar ao *paired-end* (Figura 10) e se diferencia apenas por possuir um espaçamento maior entre as leituras, ou seja, apresenta uma distância maior entre as leituras. É utilizado quando se deseja obter bibliotecas com fragmentos (insertos) mais longos. A saída desse tipo de leitura são dois arquivos pareados (METZKER, 2010).

[15] https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html

Figura 11 – Ilustração de leituras *Mate-Pair*



Fonte: Adaptado e traduzido de Illumina[16]

2.3 Arquivo FASTQ

Atualmente, os arquivos gerados por esses sequenciadores após o processo de sequenciamento possuem um formato padrão FASTQ. Nesses arquivos, cada leitura é composta por quatro linhas que além das bases sequenciadas possuem um valor de qualidade atrelado a cada base sequenciada.

O valor de qualidade é representado através da escala Phred[17], contudo com intuito de reduzir o tamanho do arquivo (tendo em vista que o padrão anterior produzia dois arquivos: um arquivo contendo as sequências no padrão FASTA e outro contendo apenas a qualidade em valores decimais com a extensão QUAL), os valores de qualidade são codificados no mesmo arquivo utilizando a tabela ASCII. Com isso, utiliza-se apenas um caractere para representar a qualidade atrelada a cada base sequenciada e não dois caracteres como no formato anterior, que utilizava valores decimais. A Figura 12 abaixo mostra um exemplo de leitura no formato padrão FASTQ.

[16] https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html
[17] É uma medida para identificar a qualidade das sequencias (A, T, G, C) identificadas pelos sequenciadores de DNA (Ewing *et al*., 1998).

Figura 12 – Exemplo de uma leitura no formato FASTQ

```
@M00707:80                  # Cabeçalho '@'
GGATTGTAGAACTTTGGAATCAAAATCGGA  # Sequencia de DNA
+                           # Separador '+', indica o início da qualidade
CCC%#%GGGG%$#@GGGG%#@!GGGGGGC  # Score de Qualidade - Phred-Scaled
```

Fonte: Elaborada pelo autor

2.4 *Strings* e redundância de *strings*

Em ciência da computação, uma sequência de caracteres é chamada de *string*. Por exemplo, “esta frase é uma *string*”. Uma linguagem é um conjunto finito ou infinito de *strings*. Em bioinformática uma sequência de DNA é identificada como uma “*string*”. Este é um exemplo de uma terminologia compartilhada por estas duas áreas (TISDALL, 2009).

Nos laboratórios de biologia molecular uma das tarefas mais comum é a concatenação de fragmentos de DNA por similaridade (TISDALL, 2009). Assim, a maioria dos algoritmos desenvolvidos para análises ômicas são capazes de lidar com esta operação. Concatenar duas *strings* significa unir o conteúdo das mesmas (TISDALL, 2009). Por exemplo, considerando os fragmentos de DNA ‘AGACGTCCG’ e ‘CCGGTTTGCA’ a concatenação dessas duas *strings* gera ‘AGACGTCCGGTTTGCA’.

Além da concatenação precisamos entender quando ocorre a redundância de *strings*. Redundância é quando uma mesma informação é armazenada repetidamente de forma desnecessária dentro de um conjunto de dados (HEUSER, 2009). Redundância de *strings* em dados NGS ocorre quando a mesma sequência de DNA (*string*) é repetida uma ou mais vezes dentro do conjunto de dados (TISDALL, 2009). Por exemplo:

Seq¹: GCTTTGGGGGCCAAAATTTT

Seq²: GCTTTGGGGGCCAAAATTTT

Essa redundância de leituras aumenta o volume de dados gerado pelos sequenciadores NGS, o que eleva o custo computacional para processamento destes dados, principalmente quando se trata de consumo de memória necessário para o manuseio em análises futuras como, por exemplo, na montagem do genoma destes organismos (EBBERT *et al*., 2016).

Existem diversos algoritmos que auxiliam na busca e remoção de redundância de *strings* em grandes volumes de dados. Um algoritmo bastante utilizado é a função *hash*. Função *hash* é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo (KONHEIM, 2010). Uma aplicação deste algoritmo é a tabela *hash,* uma estrutura de dados especial, que associa chaves de pesquisa *(hash)* a valores (GOODRICH & TAMASSIA, 2013). A grande vantagem da utilização dessa estrutura está no desempenho, pois é possível efetuar consultas rápidas em grande volume de dados aplicando a função *hash* no momento de armazenar e no momento de buscar a chave. Um exemplo de aplicação destas funções em bioinformática é na busca de trechos similares em sequências de DNA.

2.5 Remoção de redundância em dados NGS

Como mencionado no item 1.4, durante a fase de amplificação do DNA ocorre a super-representação dos dados, esta fase resulta na redundância de leituras no conjunto de dados gerado pelos sequenciadores NGS.

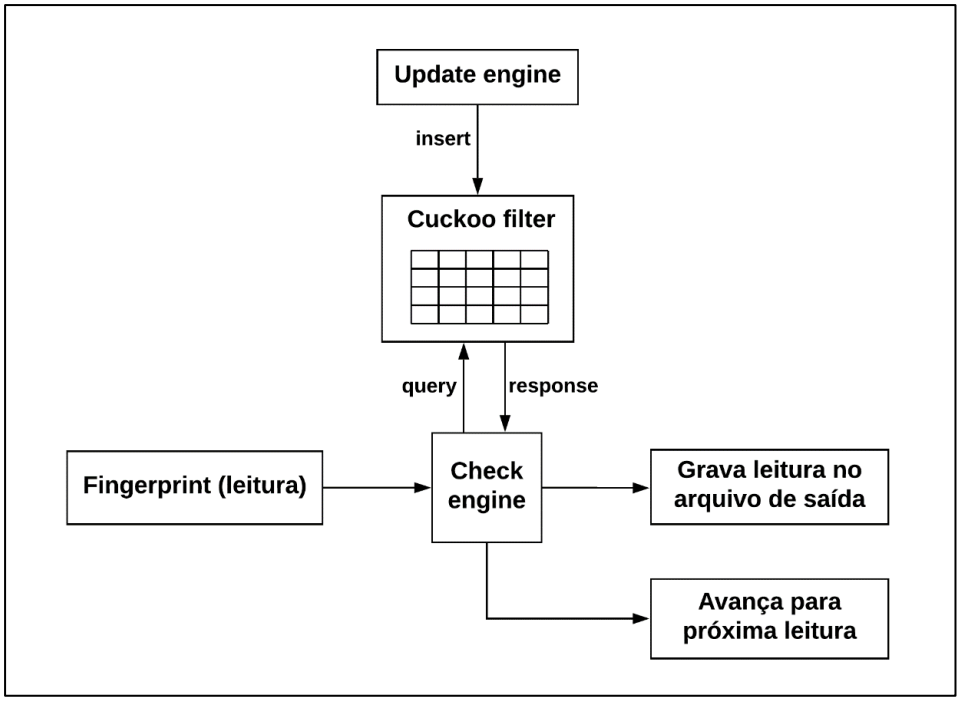
Para realizar a remoção dessas leituras duplicadas foi utilizada neste trabalho a estrutura de dados probabilística *Cuckoo Filter* (FAN *et al.*, 2014) usada para consultas de associação de conjunto aproximados de forma rápida e eficaz. Desenvolvido por Fan *et al.* (2014), o *Cuckoo Filter* surgiu como um aperfeiçoamento ao *Bloom Filter* (BLOOM, 1970), introduzindo o suporte à adição e exclusão dinâmica de itens, melhora no desempenho de pesquisas e melhor eficiência de espaço para aplicações com baixa taxa de falsos-positivos.

O *Cuckoo Filter* utiliza o *cuckoo hashing* (PAGH & RODLER, 2004) para resolver colisões e consiste basicamente em uma tabela compacta de *cuckoo hash* que armazena as *fingerprints* dos itens inseridos. Cada *fingerprint* é uma cadeia de

bits derivada do *hash* do item a ser inserido. Uma tabela de *cuckoo hash* consiste em uma matriz bi-dimensional onde as linhas correspondem às unidades associativas chamadas de *buckets* (depósitos) e suas células são chamadas de *slots* (entradas). Um *bucket* pode conter vários *slots e* cada *slot* é utilizado para armazenar uma única *fingerprint* de tamanho pré-definido (FAN *et al.*, 2014). Por exemplo, um *Cuckoo Filter* (2,4) possui *slots* que armazenam *fingerprints* de 2 bits e cada *bucket* da tabela pode armazenar até 4 *fingerprints*.

No processo de remoção de redundância (Figura 13) é gerada, para cada leitura, uma *fingerprint* e verificado se a mesma está contida na *engine* (C*uckoo Filter)* da ferramenta*.* Caso a resposta seja *falso*, a *fingerprint* é inserida no filtro e a leitura é armazenada em um arquivo de texto, caso contrário a leitura é descartada. Vale destacar que estas estruturas probabilísticas (*Cuckoo Filter* e *Bloom Filter*) não fornecem falsos negativos o que permite maior eficiência na remoção das leituras duplicadas do arquivo bruto. A seguir é descrito como são realizadas as operações de inserção e consultas que são utilizadas no processo de remoção de redundância.

Figura 13 - Arquitetura da abordagem da ferramenta



Fonte: Adaptado de Al-hisnawi & Ahmadi. 2017.

Para inserir um novo item na tabela o *Cuckoo Filter* utiliza a técnica *partial-key cuckoo hashing* (PAGH & RODLER, 2004). Esta técnica armazena apenas a *fingerprint* de cada leitura inserida, o que permite economizar espaço. Neste processo são utilizadas duas funções *hash* para calcular o índice de dois *buckets* candidatos. Caso algum *slot* desses *buckets* esteja vazio, a *fingerprint* da leitura é armazenada e a operação é finalizada, caso contrário o *cuckoo hashing* (PAGH & RODLER, 2004) remove uma *fingerprint* de um dos *buckets* candidatos e armazena a nova *fingerprint* nesse novo *slot*. Para a *fingerprint* removida é necessário encontrar um novo *bucket,* para isso o índice é calculado passando como entrada o *bucket* atual e sua *fingerprint.* Caso encontre um *slot* vazio, o item é armazenado e o processo é finalizado, caso contrário o processo de remover e encontrar um novo *bucket* para a *fingerprint* removida é novamente executado. Esta técnica além de economizar espaço permite a inserção e exclusão de itens de forma dinâmica. O Algoritmo 1 mostra como o *Cuckoo Filter* adiciona dinamicamente novos elementos usando o *partial-key cuckoo hashing*.

Algoritmo 1: Inserir(x)

```
1:  Begin
2:  f = fingerprint(x);
3:  h1 = hash(x);
4:  h2 = h1 ⊕ hash(f);
5:  if bucket[h1] ou bucket[h2] possui uma entrada vazia  then
6:      adiciona f para bucket[h1] ou bucket[h2];
7:      return Concluído;
8:  end IF
9:  // Deve realocar  itens existentes se não houver entradas vazias
10: h = escolher aleatoriamente h1 or h2;
11: for n = 0; n < MaxNumKicks; n++ do
12:     seleciona aleatoriamente uma entrada e do bucket[h];
13:     trocar f e a fingerprint armazenada na entrada e;
14:     h = h ⊕ hash(f);
15:     if bucket[h] possui uma entrada vazia then
16:         adiciona f em bucket[h];
17:         return Concluído;
```

18: **end IF**
19: **end for**
20: // Tabela de *cuckoo hash* é considerada cheia
21: **return** Falha;
22: **End**

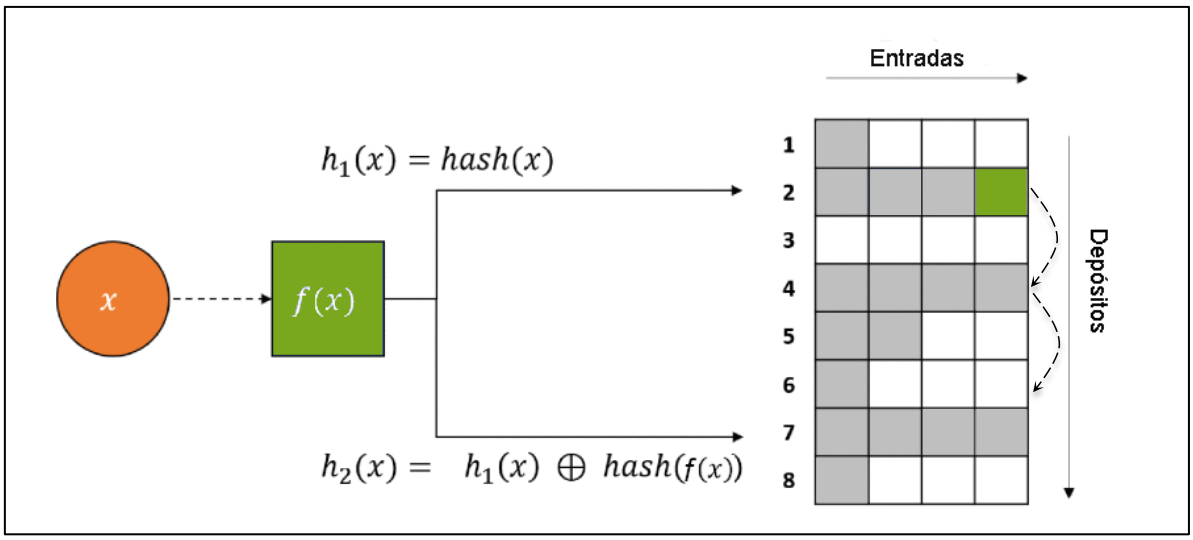
Para consultar se a *fingerprint de* uma leitura já foi inserida na tabela de *cuckoo hash*, primeiro calcula-se a *fingerprint* desta leitura e seus dois *buckets* candidatos. Após esta etapa, os dois *buckets* são lidos e caso encontre na tabela uma *fingerprint* igual ao da leitura pesquisada o *Cuckoo Filter* retornará *True*, caso contrário retornará *False*. Esta forma de pesquisa proporciona uma melhor performance pois a busca é realizada apenas em dois *buckets* da tabela de *cuckoo hash*. O Algoritmo 2 mostra o processo de busca do *Cuckoo Filter*.

Algoritmo 2: Pesquisar(x)

1: **Begin**
2: *f = fingerprint(x);*
3: h_1 *= hash(x);*
4: $h_2 = h_1 \oplus hash(f);$
5: **if** *bucket*[h_1] or *bucket*[h_2] tem *f* **then**
6: **return** True;
7: **end if**
8: **return** False;
9: **End**

Abaixo a Figura 14 mostra a inserção de um item x no *Cuckoo Filter* onde é armazenado apenas a *fingerprint* do item x inserido. A função *hash()* é usada para calcular o índice do *bucket* candidato e *f()* é usada para gerar a *fingerprint.* O *bucket* alternativo é calculado com base no índice do *bucket* atual e no valor da *fingerprint* armazenada.

Figura 14 – Representação de uma inserção usando *Cuckoo Filter*



Fonte: Adaptado e traduzido de Grashöfer *et al*. 2018.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos relacionados à ferramenta proposta, os quais foram previamente selecionados por meio de uma revisão sistemática da literatura. Procuramos destacar as principais características, funcionalidades e principalmente como essas ferramentas realizam a remoção de leituras duplicadas em dados gerados por sequenciadores de segunda geração.

3.1 CD-HIT

Lançado em 2006 (LI & GODZIK, 2006), o programa CD-HIT é um pacote de software gratuito escrito em C++ e executado através de linha de comando para agrupar e comparar sequências de DNA. É utilizado na bioinformática para reduzir os esforços computacionais em muitas tarefas de análises posteriores do organismo sequenciado como, por exemplo, a montagem *de novo*. O CD-HIT utiliza estratégia livre de alinhamento[18] para identificar e remover duplicados em leituras curtas produzidas por sequenciadores NGS.

O CD-HIT é uma abordagem de algoritmos de agrupamento incremental. Primeiro o algoritmo classifica as leituras em ordem decrescente de comprimento e as processa do maior para o menor. A primeira leitura é automaticamente classificada como a sequência representante do primeiro *cluster*. Em seguida, cada leitura restante é comparada com as sequências representantes encontradas antes dela e é classificada como redundante ou representante com base em sua semelhança com as sequências encontradas.

A ferramenta CD-HIT-DUP identifica e remove redundância de leituras fragmentos e pareadas geradas com a plataforma Illumina. Enquanto que o *script* CD-HIT-454 é utilizado para identificar e remover duplicações de leituras geradas pela plataforma 454.

[18] Não necessita mapear as leituras com uma sequência de referência para identificar e remover as leituras duplicadas.

3.2 FastUniq

A ferramenta FastUniq foi projetada para remoção de leituras duplicadas de final pareado (XU *et al*., 2012). O FastUniq utiliza a estratégia *de novo* e não requer, como pré-requisito, sequências genômicas de referência para identificar leituras duplicadas. A remoção é realizada em um processo de três etapas: inicialmente, todas as leituras pareadas são carregadas na memória; então, os pares lidos são classificados e, finalmente, as sequências duplicatas são identificadas comparando os pares de leitura adjacentes na lista classificada.

Na etapa de importação de leituras é criada uma arquitetura com três camadas. O objeto da camada mais baixa *'fastq'* armazena os dados de cada leitura. O objeto da camada intermediária *'fastq_pair'* armazena os dados de cada par de leitura e é composto por dois objetos *'fastq'*. E o objeto da camada mais alta é composto por um grande número de objetos *'fastq_pair'*. Depois da importação, a lista de *'fastq_pair'* é indexada para acesso rápido a qualquer objeto da lista.

Na etapa de classificação, o FastUniq utiliza o algoritmo *merge sort* para classificar todos os objetos *'fastq_pair'* da lista. Primeiro ele compara as sequências da primeira leitura e só realiza a comparação das sequências da segunda leitura se as da primeira forem iguais. Na última etapa, a remoção é realizada através da identificação de duplicados na lista classificada de *'fastq_pair'*, comparando os pares de leitura adjacentes da lista.

O FastUniq foi escrito utilizando a linguagem C e pode ser executado através de linha de comando na maioria dos sistemas operacionais compatíveis com UNIX/Linux.

3.3 Clumpify

Clumpify (BUSHNELL, 2016) é uma ferramenta escrita em Java que não utiliza alinhamento ou mapeamento para remoção de redundância de leituras fragmentos ou pareadas. O Clumpify primeiro reordena o conjunto de dados para que leituras semelhantes fiquem próximas. Após a reordenação, são criados grupos de leituras

que se sobrepõem resultando em uma lista de grupos de leituras classificadas. Ao final, o processo de remoção é feito compactando os grupos de leituras gerando um único arquivo sem leituras duplicadas.

O Clumpify foi projetado para dados de sequenciadores Illumina, Ion Torrent ou Pacbio. O Clumpify não funciona bem com leituras que tenham cobertura muito baixa (Ex. 1x) ou com dados que possuem elevada taxa de erro.

3.4 ParDRe

O ParDRe (GONZÁLEZ-DOMINGUEZ *et al*., 2016) é uma ferramenta que utiliza técnicas de processamento paralelo para remover leituras duplicadas produzidas por sequenciadores NGS, com suporte para conjunto de dados fragmentos ou pareados. Baseado na estratégia *de novo,* só precisa dos dados de entrada do NGS para remover as leituras duplicadas.

O ParDRe utiliza uma abordagem bit a bit para comparar as sequências de DNA e emprega *multithreading* e *Message Passing Interface* (MPI) para explorar o poder computacional dos sistemas *multicore* atuais. O procedimento começa agrupando todas as leituras com o mesmo prefixo que são armazenadas em *clusters*. Cada processo MPI fica responsável por vários *clusters.* Após o término do agrupamento, o ParDRe compara os sufixos das leituras que estão no mesmo *cluster* e remove as leituras que possuem sufixos iguais. Depois que todos os *clusters* forem analisados, esta análise é feita em modo paralelo, as leituras restantes de cada *cluster* são gravadas em arquivos temporários. Ao final do processo o ParDRe concatena todas as leituras em um único arquivo e exclui os arquivos temporários.

3.5 MarDRe

Desenvolvida utilizando a linguagem de programação Java, o MarDRe (EXPÓSITO *et al*., 2017) é uma ferramenta baseada no *MapReduce* para remover redundância de leituras fragmentos ou pareadas.

Assim como na ferramenta ParDRe, as leituras com o mesmo prefixo são agrupadas em *clusters* para em seguida seus sufixos serem comparados. Os *clusters* são gerados em paralelo durante a fase de mapeamento e as leituras do mesmo *cluster* comparadas durante a fase de redução. As leituras não duplicadas são gravadas em arquivos temporários utilizando o sistema de armazenamento do *Hadoop Distributed File System* (HDFS). Após o término da tarefa do MapReduce, é executada uma operação no nível do HDFS para mesclar todos os arquivos temporários em um único arquivo com as leituras.

As etapas para remoção de leituras duplicadas do MarDRe e ParDRe são iguais, porém, devido às técnicas computacionais distintas aplicadas a cada uma das ferramentas, o MarDRe possui melhor desempenho em sistemas distribuídos, enquanto que o ParDRe tem desempenho melhor em sistemas *multicore* (EXPÓSITO *et al*., 2017).

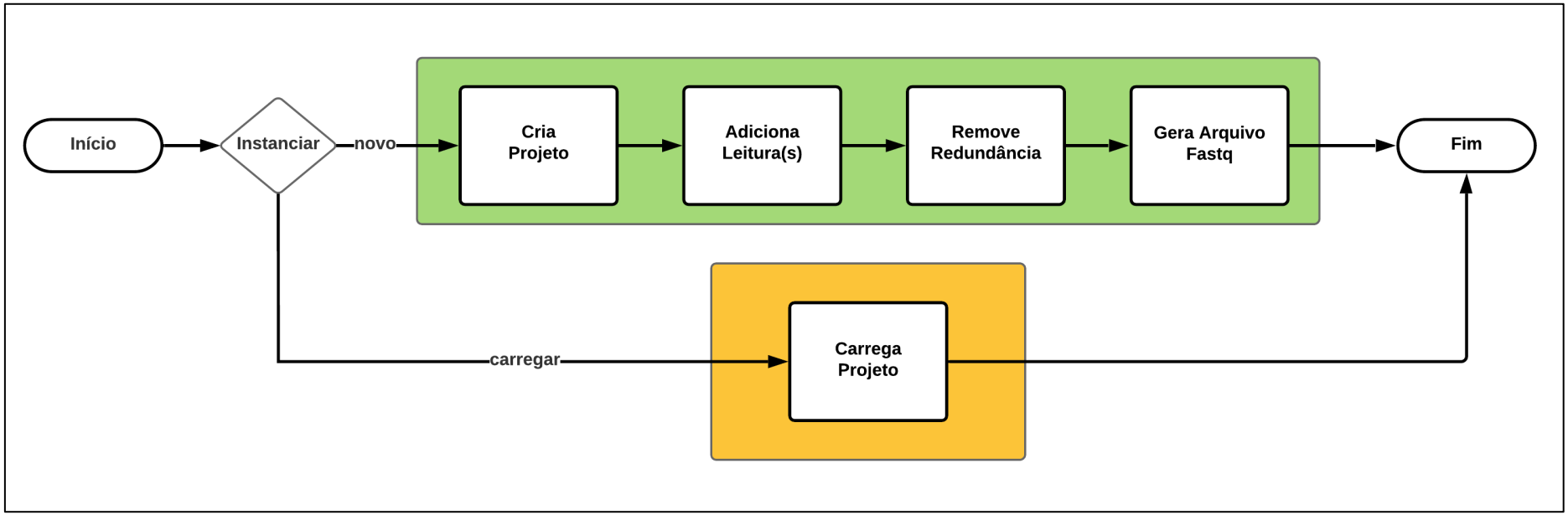
4 PROPOSTA DA FERRAMENTA

A ferramenta proposta foi projetada e otimizada para remover leituras duplicadas de grandes conjuntos de dados oriundos de sequenciadores NGS. Possui independência de plataforma, ou seja, processa leituras de qualquer plataforma NGS, e aceita como entrada arquivos de sequência no formato FASTQ (padrão das plataformas NGS) de leituras fragmentos ou pareadas.

Desenvolvida em JAVA, a ferramenta pode ser executada em sistemas operacionais UNIX/Linux e Windows e a mesma possui duas versões: uma que pode ser executada em linha de comando e outra com interface gráfica. Sendo que, a versão com interface gráfica dispõe de um banco de dados (SQLite) para armazenar informações dos projetos dos usuários. Este armazenamento permite que ao executar o programa, o usuário possa escolher entre criar um novo projeto ou carregar um projeto criado anteriormente. Isto torna o uso da ferramenta bem simples e intuitivo, além de não necessitar a instalação de nenhuma dependência para sua execução.

O *pipeline* da ferramenta é mostrado na Figura 15 e nele são descritos todos os passos necessários no processo de remoção. A seguir é detalhada a arquitetura da ferramenta, suas funcionalidades, diagrama de casos de uso e diagrama de entidade relacionamento utilizados na sua construção.

Figura 15 - Representação do pipeline da ferramenta versão com interface gráfica



Fonte: Elaborada pelo Autor

4.1 Arquitetura da ferramenta

A ferramenta proposta utiliza uma arquitetura em três camadas, essas camadas são compostas por módulos que implementam as funcionalidades do sistema. A seguir detalharemos cada camada que compõe a ferramenta.

4.1.1 Camada de Apresentação

Na camada de apresentação ou interface do usuário estão todas as interações do usuário com a ferramenta. Esta camada também permite ao usuário acompanhar visualmente o *status* do processamento de seu projeto pela tela Log ou por uma barra circular que exibe o progresso de processamento.

4.1.2 Camada de Negócio

Também chamada de camada de lógica de negócio, é a camada onde estão todos os módulos responsáveis por processar e gerar os resultados dos dados do usuário. Nesta camada encontra-se o módulo para remoção de redundância responsável por identificar e remover as leituras duplicadas do conjunto de dados bruto. Também nesta camada está o módulo responsável pela geração do arquivo com as leituras únicas do *dataset*.

4.1.3 Camada de Persistência

A camada de persistência é composta por um banco de dados (SQLite) responsável por armazenar os projetos criados pelos usuários e as informações que são geradas de forma automática pelo sistema. Esta camada permite que, ao inicializar a ferramenta, o usuário possa criar um novo projeto ou carregar um projeto criado anteriormente, permite também o acompanhamento do *status* de processamento de cada projeto e, caso ocorra algum tipo de erro ou se o usuário optar por interromper o processamento, ele poderá ser retomado posteriormente.

4.2 Funcionalidades da ferramenta

Nesta seção serão apresentadas as funcionalidades da ferramenta descrevendo seus requisitos funcionais e não funcionais.

4.2.1 Requisitos Funcionais

Gerenciar Projetos:

A ferramenta proposta conta com um banco de dados embutido (SQLite) para gerenciar os projetos dos usuários. Assim, ao inicializar, o usuário tem a opção de cadastrar um novo projeto ou carregar um projeto cadastrado anteriormente. O banco de dados também possibilita o controle do *status* de processamento do projeto em execução, portanto, se ocorrer algum tipo de erro ou se o usuário optar por interromper o processamento ele poderá ser retomado posteriormente.

Remover Redundância de Leituras:

A remoção de leituras duplicadas busca reduzir os requisitos de memória e o tempo computacional de análises como a montagem *de novo* em conjunto de dados NGS. Para tanto, nesta proposta se utiliza uma estrutura probabilística (*Cuckoo Filter*) eficiente em identificação e remoção de *strings* duplicadas em grande volume de dados.

Gerar Arquivo sem Redundância:

Para atender a este requisito ao término do processamento dos dados é gerado um arquivo (fragmentos) ou dois arquivos (pareados) no formato FASTQ com as leituras únicas do *dataset*.

4.2.2 Requisitos Não Funcionais

Usabilidade:

As ferramentas disponíveis são executadas em linha de comando e exigem conhecimento avançado em computação para sua instalação e em alguns casos na instalação de suas dependências. Isto acaba impactando na produção de

pesquisadores, grupos de pesquisa e laboratórios que não contam com bioinformata residente para operar estes programas. A ferramenta proposta possui uma versão com interface gráfica onde toda a interação no processo de remoção de redundância é realizada através de interface gráfica e esta deverá se comportar da mesma maneira, independente do sistema operacional em que esteja sendo executada.

Desempenho:

As plataformas NGS geram um grande volume de dados e isso aumenta o custo computacional para processar análises futuras destes dados. Por isso, a ferramenta proposta implementa otimizações e técnicas computacionais que buscam reduzir o consumo computacional destas análises.

Compatibilidade:

Buscando atender tanto usuários de sistemas operacionais UNIX/Linux quanto de Windows, a ferramenta deverá ser compatível com esses sistemas operacionais e o comportamento deve ser o mesmo, tanto no que se refere às funcionalidades quanto à sua execução.

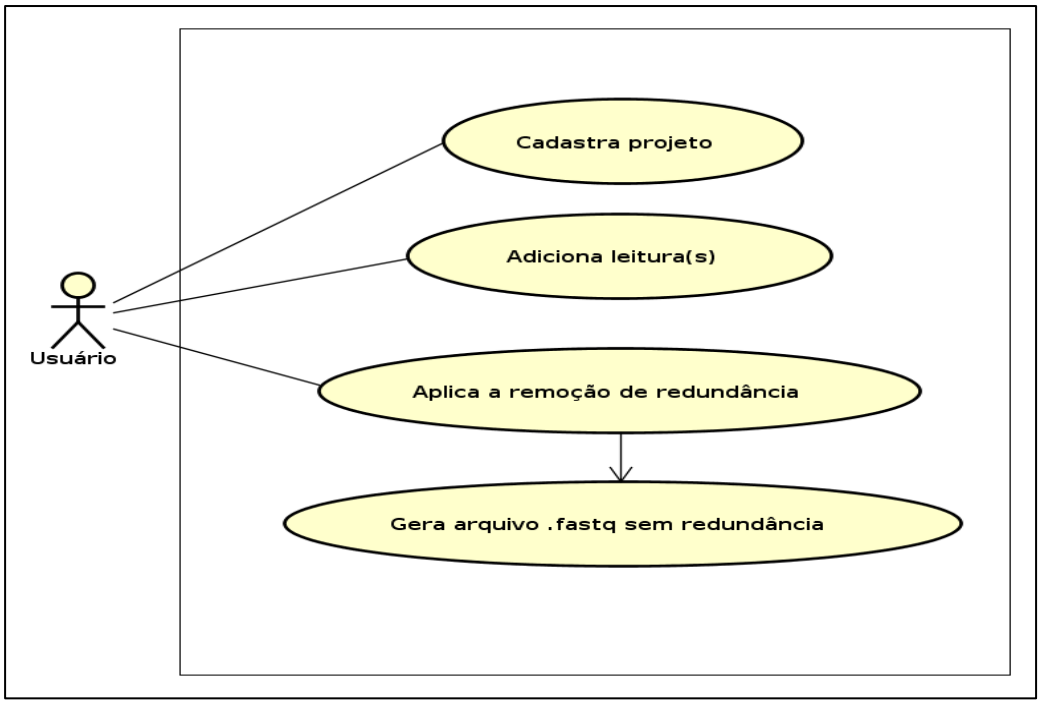
4.3 Casos de uso da ferramenta

A seguir serão apresentados os diagramas de caso de uso da ferramenta. Este diagrama mostra as funcionalidades do sistema e como são utilizadas pelos usuários.

Nas Figuras 16 e 17 são demonstradas as ações do usuário para remoção de leituras duplicadas utilizando as versões com e sem interface gráfica da ferramenta. Na versão com interface (Figura 16) o usuário deve primeiramente cadastrar um projeto, em seguida adicionar a(s) leitura(s) fragmentos ou pareadas no formato FASTQ e por fim aplicar a remoção de leituras duplicadas existentes no *dataset*. Na versão sem interface (Figura 17) o usuário executa a ferramenta passando a(s) leitura(s) para aplicar a remoção de redundância. Em ambas as versões ao final do

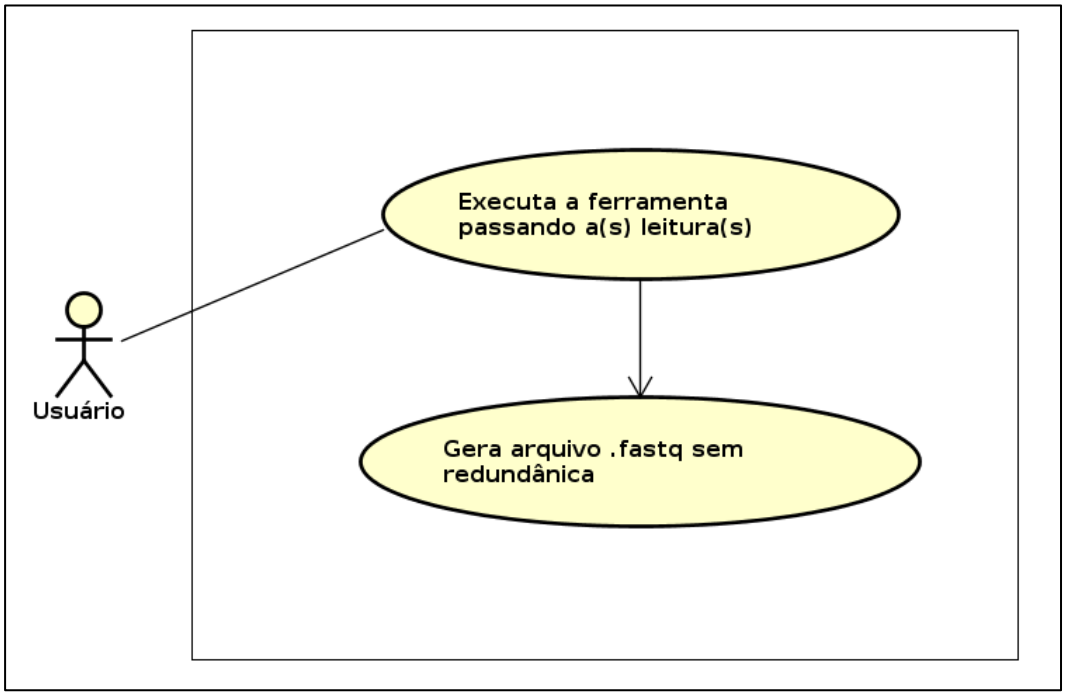
processamento é gerado um ou dois arquivos no formato FASTQ com as leituras únicas do *dataset*.

Figura 16 – Caso de uso para remoção de leituras duplicadas utilizando a versão gráfica da ferramenta



Fonte: Elaborada pelo autor

Figura 17 – Caso de uso para remoção de leituras duplicadas utilizando a versão sem interface gráfica da ferramenta

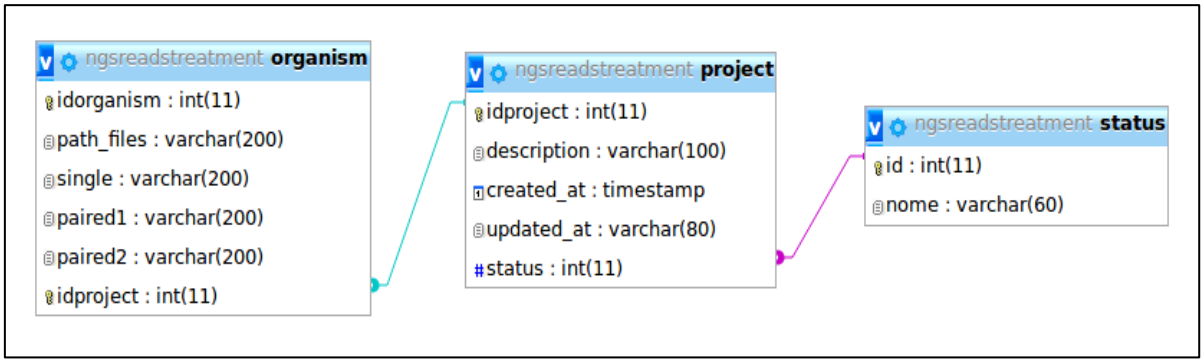


Fonte: Elaborada pelo autor

4.4 Diagrama Entidade Relacionamento da ferramenta

Neste tópico será apresentado o diagrama entidade relacionamento (DER) da ferramenta. Este diagrama apresenta as tabelas e seus relacionamentos do banco de dados utilizado pelo sistema. Abaixo a Figura 18 exibe o DER da ferramenta.

Figura 18 – Diagrama Entidade Relacionamento da ferramenta



Fonte: Elaborada pelo autor

As funções de cada tabela do diagrama de entidade relacionamento são apresentadas a seguir.

Tabela project: Esta tabela é responsável por armazenar os projetos dos usuários e o status do projeto.

Tabela status: Tabela que representa os diversos status que um projeto pode assumir.

Tabela organism: Tabela que armazena os dados do organismo relacionado a um projeto.

5 MATERIAIS E MÉTODOS

Neste capítulo serão apresentadas as fontes de dados que foram utilizadas para a validação da ferramenta, bem como a metodologia utilizada para seu desenvolvimento. Também serão descritos as ferramentas e plataformas utilizadas na construção da aplicação.

5.1 Fonte de dados

Para validação da ferramenta, foram utilizados conjuntos de dados reais e simulados. A fonte de dados de organismos reais foi o banco de dados do NCBI *(National Center for Biotechnology Information - https://www.ncbi.nlm.nih.gov),* onde estão armazenados organismos modelos para diversos experimentos biológicos e disponíveis para download de toda a comunidade científica.

Nesta etapa foram utilizados dezesseis organismos que estão listados na Tabela 1 com seus respectivos números de *Sequence Read Archive* (SRA), tamanho do *dataset*, tipo de leitura e plataforma de sequenciamento.

Tabela 1 – Lista de organismos reais juntamente com número SRA usado para validar a ferramenta

Organismo	Número do SRA	Tamanho do Dataset	Total de Leituras	Tipo de Leitura	Plataforma NGS
Escherichia coli RR1	SRR2014554	8192MB	24248885	Pareada	Illumina HiSeq 2000
Escherichia coli 042	ERR007646	2406MB	14110696	Pareada	Illumina Genome Analyzer
Escherichia coli P12b	SRR2000272	1350MB	2990758	Pareada	Illumina MiSeq
Escherichia coli KLY	SRR1424625	1682MB	6886668	Pareada	Illumina HiSeq 2000
Escherichia coli O25b:H4-ST131	SRR933487	1070MB	3214312	Pareada	Illumina Genome Analyzer IIx
Kineococcus rhizosphaerae DSM 19711	SRR6479489	2048MB	5641334	Pareada	Illumina HiSeq 2500

Kineococcus xinjiangensis DSM 22857	SRR6479482	2168MB	5971022	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis F11	SRR974839	1936MB	7279254	Pareada	Illumina HiSeq 2000
Mycobacterium tuberculosis XDR KZN 4207	SRR1144800	1884MB	7033428	Pareada	Illumina HiSeq 2000
Arcobacter Halophilus	SRR7587111	588MB	670813	Fragmentos	454 Titanium
Rhodopirellula báltica	SRR7819959	1984MB	3207713	Fragmentos	Ion Torrent
Escherichia coli O157:H7 in Romania	ERR2375157	1201MB	2106268	Fragmentos	Ion Torrent
Rathayibacter Tritici	SRR6799098	157MB	160403	Fragmentos	454 Junior
Salmonella entérica	SRR7905974	2990MB	163468	Fragmentos	PacBio
Staphylococcus aureus	SRR7739756	1336MB	86389	Fragmentos	Oxford nanopore MinIon
Pseudomonas Aeruginosa	ERR2162181	246MB	1146696	Fragmentos	SoliD 5500

Os dados simulados foram utilizados com o objetivo de efetuar a validação da ferramenta de maneira controlada, ou seja, na geração dos dados simulados foram controladas as taxas de duplicação de PCR e com isso foi possível quantificar a taxa de remoção de leituras duplicadas da ferramenta proposta.

A ferramenta ART versão 2.5.89 (HUANG *et al*., 2012) foi usada para gerar os conjuntos de dados simulados. Esta ferramenta gera leituras simuladas de diversas plataformas NGS, por exemplo: 454, Illumina e etc., baseadas em um arquivo de referência no formato FASTA. O ART suporta a simulação de leituras fragmentos ou pareadas simulando erros e qualidade reais de leitura do sequenciamento. Os dados simulados são essenciais para testar ou avaliar o desempenho das ferramentas desenvolvidas para análise de dados de sequenciamento NGS.

Para esta etapa da validação da ferramenta, foram simuladas leituras do sequenciamento nas plataformas Illumina HiSeq 2500 e Roche 454 GS FLX

Titanium. Primeiro foram utilizados os organismos listados na Tabela 2 e posteriormente os organismos com diferentes valores de cobertura listados na Tabela 3. Nos testes com organismos simulados com variação de cobertura as leituras foram contadas para determinar a quantidade de leituras, o número de leituras únicas e a quantidade de leituras redundantes nos dados brutos de cada conjunto de dados. O objetivo desta etapa era verificar a eficiência de remoção das ferramentas avaliadas.

Tabela 2 – Lista de dados simulados usados para validar a ferramenta

Organismo	Tamanho do Dataset	Total de Leituras	Tipo de Leitura	Plataforma NGS
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	910MB	2917800	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis KZN 4207	914MB	2929900	Pareada	Illumina HiSeq 2500
Escherichia coli O103:H2 str. 12009	1132MB	3632800	Pareada	Illumina HiSeq 2500
Arcobacter halophilus strain CCUG 53805	588MB	1875000	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	104MB	251688	Pareada	454 GLS FLX
Mycobacterium tuberculosis KZN 4207	104MB	252954	Pareada	454 GLS FLX
Escherichia coli O103:H2 str. 12009	130MB	313944	Pareada	454 GLS FLX
Arcobacter halophilus strain CCUG 53805	70MB	161900	Pareada	454 GLS FLX

Tabela 3 – Lista de dados simulados com diferentes valores de cobertura usados para validar a ferramenta

Organismo	Cobertura	Tamanho do Dataset	Total de Leituras	Tipo de Leitura	Plataforma NGS
Mycobacterium bovis BCG str. Korea 1168P	300x	2722MB	8753400	Pareada	Illumina HiSeq 2500
Mycobacterium bovis BCG str. Korea 1168P	200x	1814MB	5835600	Pareada	Illumina HiSeq 2500
Mycobacterium bovis BCG str. Korea 1168P	100x	908MB	2917800	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis KZN 4207	300x	2742MB	8789700	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis KZN 4207	200x	1828MB	5859800	Pareada	Illumina HiSeq 2500
Mycobacterium tuberculosis KZN 4207	100x	914MB	2929900	Pareada	Illumina HiSeq 2500
Escherichia coli O103:H2 str. 12009	300x	3400MB	10898400	Pareada	Illumina HiSeq 2500

Escherichia coli O103:H2 str. 12009	200x	2266MB	7265600	Pareada	Illumina HiSeq 2500
Escherichia coli O103:H2 str. 12009	100x	1132MB	3632800	Pareada	Illumina HiSeq 2500

5.2 Metodologia de desenvolvimento da ferramenta

Para a metodologia de desenvolvimento foi adotado o modelo iterativo e incremental. A ferramenta foi dividida em módulos, e a evolução ocorreu através da construção incremental e iterativa de cada novo módulo, agregando novas funcionalidades à ferramenta.

5.3 Linguagem de programação e banco de dados

A ferramenta foi desenvolvida utilizando a linguagem de programação JAVA (http://www.oracle.com/) e usada a biblioteca *Swing* para criar a interface gráfica. *Swing* é uma biblioteca oficial do JAVA que está inclusa em qualquer *Java Runtime Environment* (JRE) ou *Java Development Kit* (JDK) e uma de suas características é sua portabilidade, ou seja, independente do sistema operacional em que a ferramenta seja executada ela sempre terá a mesma interface (tamanhos, cores, etc).

O projeto foi criado e gerenciado utilizando a ferramenta Apache Maven (https://maven.apache.org). Dentre os principais recursos desta ferramenta podemos destacar: simplificação na configuração de projetos, gerenciamento de dependências de forma automatizada e a geração do *Java ARchive* (JAR) com todas as dependências utilizadas no projeto. Neste projeto, o Maven foi utilizado para gerenciamento de dependências e automação de *build*.

Para gerenciar os projetos do usuário foi utilizado o banco de dados SQLite version 3 (https://www.sqlite.org/). Escrito em linguagem C, o SQLite é uma biblioteca que implementa um banco de dados SQL embutido, ou seja, aplicações que utilizam o SQLite não necessitam instalar e executar um sistema gerenciador de banco de dados (SGBD) para ter acesso a um banco de dados SQL. O SQLite

é um banco de dados em que todas as alterações e consultas são Atômicas, Consistentes, Isoladas e Duráveis (ACID) (HEUSER, 2009). No projeto, o SQLite foi utilizado para permitir atender os requisitos de gerenciar projetos dos usuários, permitindo ao usuário acompanhar o processamento e retomar o processo em caso de falhas.

5.4 Remoção de redundância

Nesta etapa foi utilizado a estrutura probabilística *Cuckoo Filter* (FAN *et al.*, 2014) responsável por identificar e remover as leituras duplicadas presentes no arquivo bruto. Esta estrutura probabilística tem um ótimo desempenho em buscas e inserções em grandes volumes de dados, além de possuir baixo consumo de memória proporcionada pela técnica utilizada no armazenamento dos dados e descritos detalhadamente no capítulo 2.4 deste trabalho.

5.5 Download dos dados brutos

O NCBI armazena dados brutos de sequenciamento e informações de alinhamento de sequenciamento NGS. O NCBI recebe, processa e armazena milhares de arquivos de sequenciamento do mundo todo. Os dados de sequenciamento são disponibilizados para a comunidade científica para aperfeiçoar a reprodutibilidade e permitir novas descobertas comparando os conjuntos de dados.

Para efetuar o download dos arquivos brutos no formato FASTQ foi utilizado o Fastq-dump versão 2.9.2 (https://edwards.sdsu.edu/research/fastq-dump/). Esta ferramenta efetua o download e converte automaticamente arquivos brutos compactados para arquivos no formato FASTQ.

5.6 Avaliação de custo computacional

Para avaliar o desempenho computacional da ferramenta proposta foram definidas as seguintes métricas: tempo de processamento; número total (segundos) de CPU no modo kernel; número total (segundos) de CPU no modo usuário; percentual de uso de CPU; total de memória consumido e percentual de leituras duplicadas removidas. Essas métricas foram medidas pelo software *time* (http://man7.org/linux/man-pages/man1/time.1.html) do sistema operacional Linux que é utilizado para gerar estatísticas de um comando, *shell script* ou qualquer programa executado.

No *time,* o arquivo de saída é formatado usando a opção -f ou a variável de ambiente TIME. O formato do tipo *string* é interpretado de maneira igual ao printf: caracteres comuns são copiados diretamente e caracteres especiais usando \t (tab) e \n (nova linha), o sinal de porcentagem é representado por %%, caso contrário o % indica uma conversão. Os arquivos de saída das ferramentas avaliadas foram tabulados e o resultado é apresentado e discutido no capítulo 6.

5.7 Workstation

As análises foram realizadas localmente utilizando um equipamento com a seguinte configuração: processador Intel Core i7-2620M CPU 2.70GHz com quatro núcleos de processamento, HD de 324 GB e memória de 6GB. A distribuição do Linux selecionada para avaliação de desempenho foi o Ubuntu 18.04 LTS com o kernel 4.15.0.

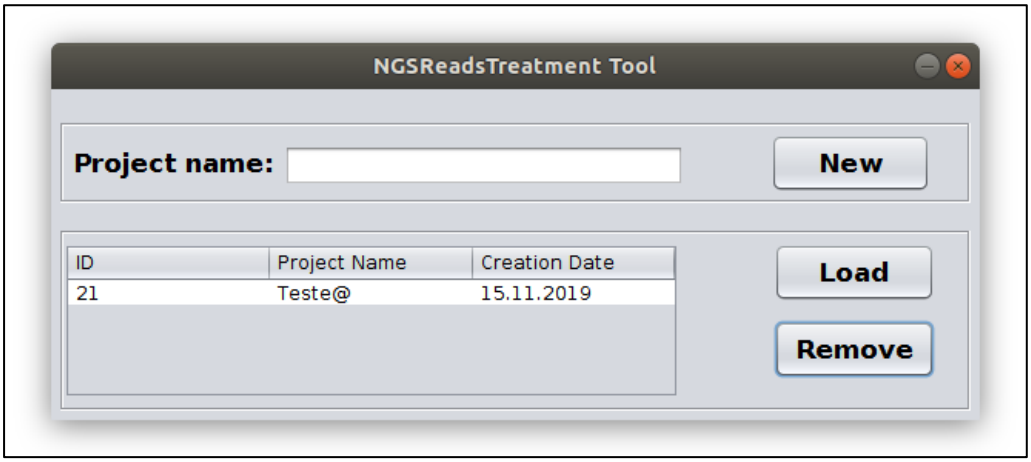
6 RESULTADOS E DISCUSSÃO

Este trabalho apresenta uma ferramenta computacional para remoção de leituras duplicadas em conjunto de dados pareados ou fragmentos. Esta ferramenta pode manipular leituras de qualquer plataforma NGS com os mesmos ou diferentes comprimentos de sequência. Usando a estrutura probabilística *Cuckoo Filter*, as leituras redundantes são identificadas e removidas comparando-se as leituras entre si. Portanto, não requer nenhum pré-requisito além do conjunto de leituras bruto geradas pelos sequenciadores NGS.

Nomeado como NGSReadsTreatment, a ferramenta possui duas versões: uma com interface gráfica e outra executada através da linha de comando. A versão com interface gráfica é orientada a projetos. Nela, o usuário deve primeiramente cadastrar o projeto e depois informar o conjunto de dados em que deseja remover a redundância de leituras, sendo possível também carregar um projeto que foi cadastrado e processado anteriormente. Esta versão ainda conta com um banco de dados embutido que possibilita o controle do *status* de processamento do projeto em execução, e caso ocorra algum tipo de erro ou se o usuário optar por interromper o processamento, ele poderá ser retomado posteriormente. A versão sem interface gráfica é disponibilizada para usuários que queiram executar a ferramenta em servidores ou que desejam integrar a mesma com outras ferramentas de bioinformática.

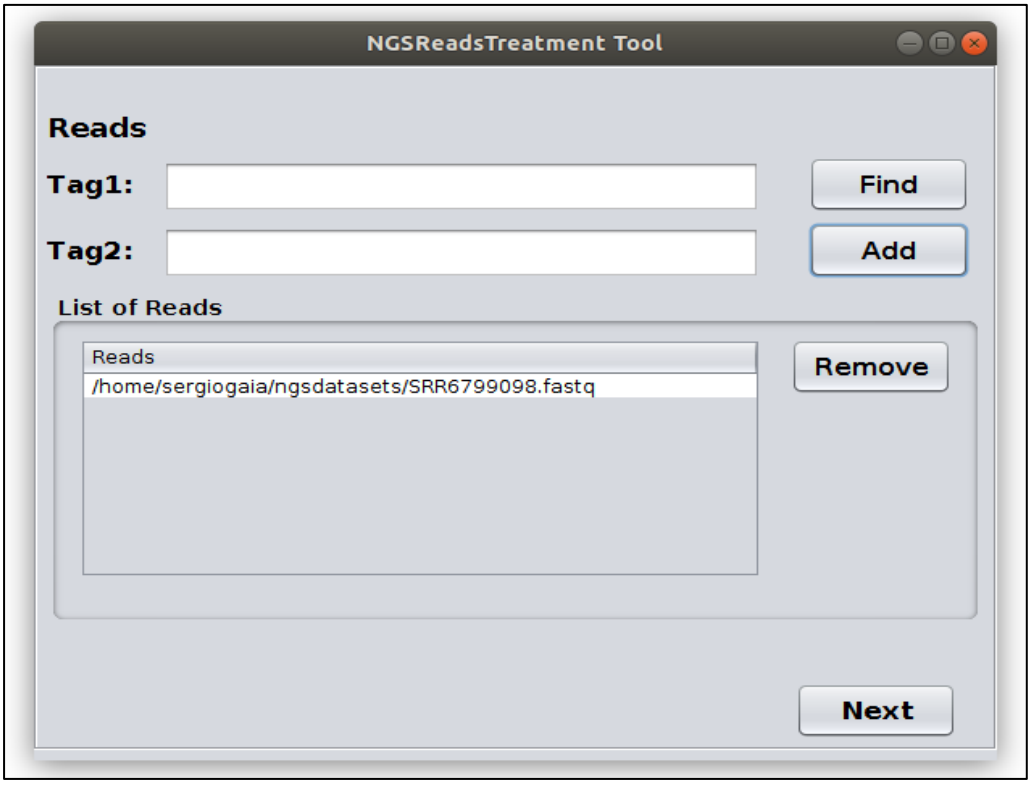
Em ambas as versões do NGSReadsTreatment é gerado ao final do processamento um arquivo (leitura fragmentos) ou dois arquivos (leituras pareadas) no formato FASTQ com as leituras únicas do *dataset* processado. As Figuras de 19 a 21 apresentam as telas da versão com interface gráfica da ferramenta.

Figura 19 – Tela de cadastro de projetos



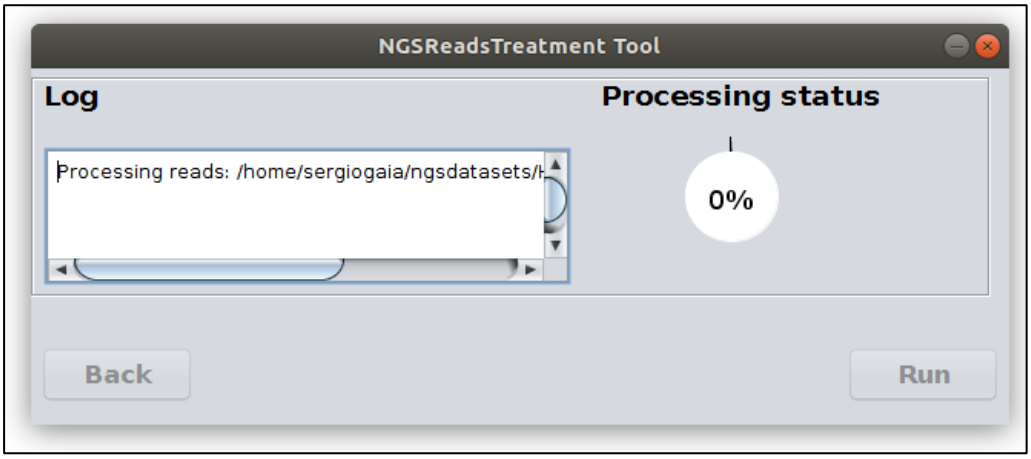
Fonte: Elaborada pelo autor

Figura 20 – Tela de entrada de leituras



Fonte: Elaborada pelo autor

Figura 21 – Tela de status do processamento



Fonte: Elaborada pelo autor

Para validar a ferramenta foram processados dois tipos de conjunto de leituras: o primeiro que utiliza dados reais de dezesseis organismos disponíveis e obtidos do banco de dados do NCBI; e um segundo que utiliza conjunto de dados simulados de oito organismos gerados pela ferramenta ART (HUANG *et al*. 2012). Os dados simulados foram utilizados para validar a ferramenta de maneira controlada e destes conjuntos de dados, três organismos (*Escherichia coli* O103: H2 str. 12009, *Mycobacterium tuberculosis* KZN 4207 e *Mycobacterium bovis* BCG str. Korea 1168P) foram simulados com diferentes coberturas (100x, 200x e 300x) de sequenciamento.

Nesta fase, foram comparados os percentuais de remoção da redundância em cada organismo (real e simulado) processado, assim como a avaliação da quantidade do total de memória utilizada por cada ferramenta avaliada. As ferramentas e suas respectivas versões avaliadas foram: FastUniq 1.1, ParDRe 2.2.5, MarDre 1.3, CD-HIT-DUP 4.6.8, Clumpify (bbmap) (https://jgi.doe.gov) e o NGSReadsTreatment.

Primeiramente foi avaliado o comportamento das ferramentas utilizando o conjunto de dados reais. O percentual de remoção da redundância em cada organismo, assim como a avaliação da quantidade do total de memória utilizada por cada ferramenta é demonstrado nas Tabelas 4 e 5, respectivamente.

Na Tabela 4 pode-se observar que o NGSReadsTreatment obteve em treze dos dezesseis organismos analisados maior percentual de leituras redundantes removidas, sendo que em um deles o percentual foi igual, ou seja, conseguiu identificar e remover a maior quantidade de redundâncias. Constatou-se também que nos organismos SRR2000272, SRR1424625 e SRR2014554 ocorreram problemas no processamento pelas outras ferramentas, dentre eles podemos citar: travamento do equipamento durante o processamento e falhas no processamento devido à existência de leituras órfãs no arquivo de leituras.

Tabela 4 – Percentual de remoção de redundância por ferramenta para cada organismo real. NP – nenhuma redução de redundância.

Organismo	Tipo de Leitura	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
SRR2014554	Pareada	NP	NP	NP	NP	NP	0.29%
ERR007646	Pareada	50.55%	50.55%	5.55%	50.55%	50.65%	50.50%
SRR2000272	Pareada	0.82%	0.81%	NP	0.81%	0.95%	1.91%
SRR1424625	Pareada	0%	0%	0%	0%	0.20%	0.94%
SRR933487	Pareada	0.72%	0.72%	0.72%	0.72%	1.11%	1.90%
SRR6479489	Pareada	0.14%	0.14%	0.13%	0.14%	0.19%	1.21%
SRR6479482	Pareada	0.14%	0.14%	0.14%	0.14%	0.18%	1.17%
SRR974839	Pareada	48.74%	48.74%	48.74%	48,74%	49.07%	49.08%
SRR1144800	Pareada	0.06%	0.06%	0.06%	0.06%	0.10%	0.94%
SRR7587111	Fragmentos	NP	0.37%	0.37%	0.37%	0.43%	0.87%
SRR7819959	Fragmentos	NP	0.74%	0.74%	NP	0.80%	1.36%
ERR2375157	Fragmentos	NP	0.07%	0.07%	NP	0.08%	1.55%
SRR6799098	Fragmentos	NP	2.11%	2.11%	2.11%	2.22%	2.22%
SRR7905974	Fragmentos	NP	0%	NP	NP	0%	0.13%
SRR7739756	Fragmentos	NP	0%	NP	NP	0%	0.08%
ERR2162181	Fragmentos	NP	0%	0%	0%	NP	0.33%

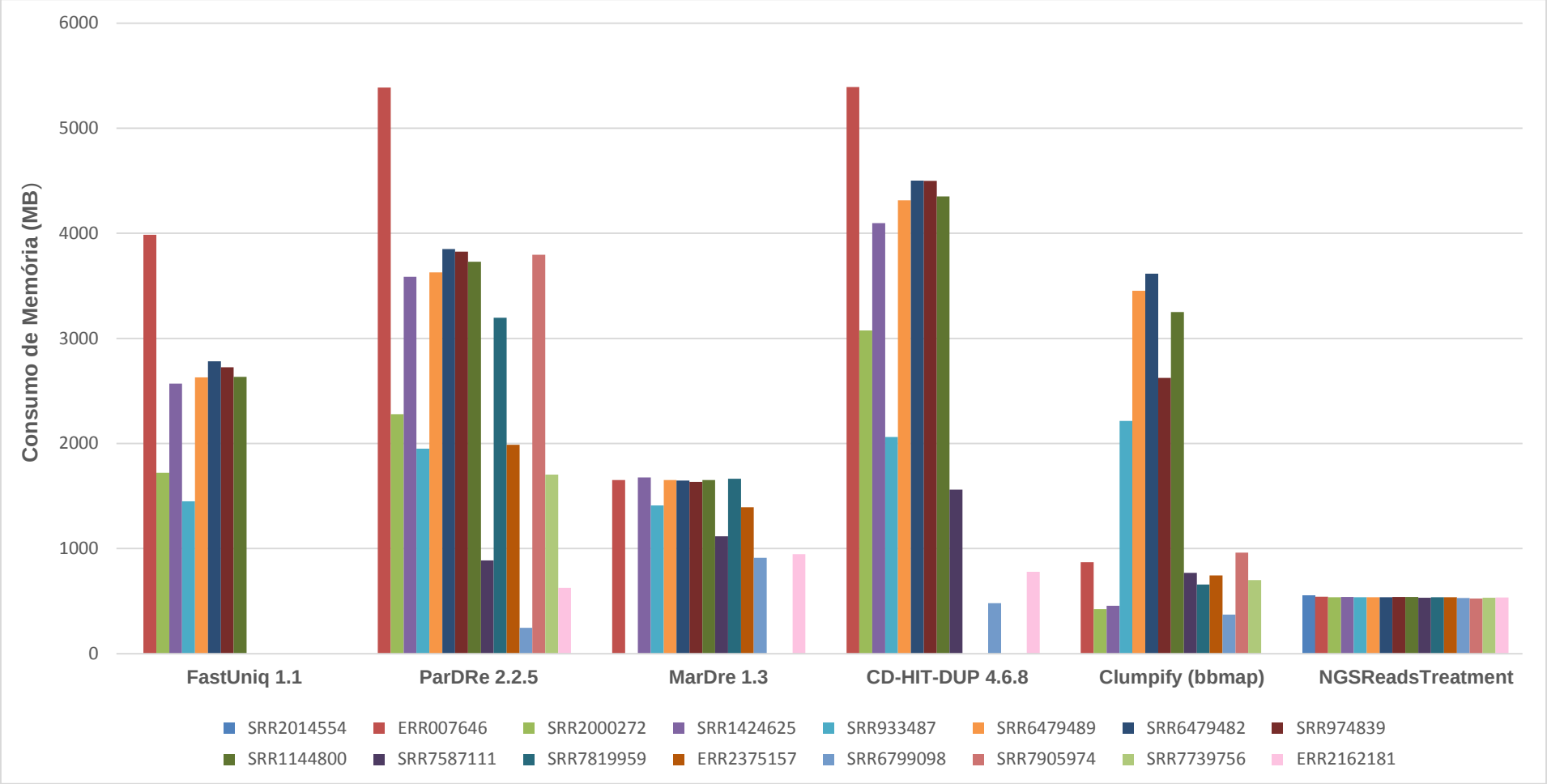
O NGSReadsTreatment foi a única ferramenta que obteve êxito no processamento do organismo SRR2014554 que possui tamanho de 4GB. Como observado na Tabela 4, todas as demais ferramentas testadas apresentaram erro no processamento das leituras deste organismo. O resultado comprova a eficiência

da ferramenta no processamento de remoção de leituras duplicadas em grande volume de dados.

A Tabela 5 mostra a quantidade de memória total utilizada por cada ferramenta no processamento das leituras brutas. Observa-se que assim como na Tabela 4 existem organismos que apresentaram problemas na execução pelas outras ferramentas, são eles: SRR2000272 pela existência de leituras órfãs no arquivo bruto e o SRR2014554 por se tratar de um arquivo de 4GB.

A ferramenta FastUniq não utiliza leituras fragmentos em suas análises, assim, não foi possível realizar o processamento do conjunto de dados de leituras deste tipo com a ferramenta. Porém, em todos os casos foi possível a utilização do NGSReadsTreatment, demonstrando sua eficiência no uso da memória, pois foi o que menos utilizou memória dentre todas as ferramentas na maioria das análises, conforme demonstrado na Figura 22 a seguir.

Figura 22 – Quantidade de uso de memória por cada ferramenta computacional para processamento de organismos reais.



Fonte: Elaborada pelo autor

Tabela 5 – Quantidade de memória utilizada por ferramenta em MB para cada conjunto de dados reias. NP - Não Processado devido a erros.

Organismo	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
SRR2014554	NP	NP	NP	NP	NP	549
ERR007646	3987	5387	1653	5393	870	543
SRR2000272	1722	2278	NP	3076	423	537
SRR1424625	2571	3586	1676	4097	455	539
SRR933487	1449	1950	1411	2063	2215	538
SRR6479489	2629	3629	1652	4313	3454	538
SRR6479482	2783	3850	1647	4501	3616	538
SRR974839	2725	3825	1634	4499	2625	540
SRR1144800	2633	3730	1653	4351	3250	540
SRR7587111	NP	888	1118	1561	769	533
SRR7819959	NP	3197	1665	NP	659	537
ERR2375157	NP	1989	1394	NP	744	537
SRR6799098	NP	247	913	481	373	531
SRR7905974	NP	3796	NP	NP	961	526
SRR7739756	NP	1704	NP	NP	700	532
ERR2162181	NP	625	947	779	NP	535

Buscando melhorar a validação do NGSReadsTreatment, as mesmas análises realizadas com os conjuntos de dados reais (dezesseis organismos) foram realizadas com conjuntos de dados simulados (oito organismos) gerados pela ferramenta ART (HUANG *et al*., 2012). O objetivo desta etapa era efetuar a validação do NGSReadsTreatment de maneira controlada.

Observou-se que assim como nos testes com organismos reais o NGSReadsTreatment foi eficiente também com organismos simulados, conforme mostrado nas Tabelas 6 e 7.

Tabela 6 – Percentual de remoção de redundância por ferramenta para cada organismo simulado. NP – nenhuma redução de redundância.

Organismo	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	0.08%	NP	0.08%	0.08%	0.20%	0.77%
Mycobacterium tuberculosis KZN 4207	0.08%	NP	0.08%	0.08%	0.20%	0.05%
Escherichia coli O103:H2 str. 12009	0.09%	NP	0.09%	0.09%	0.23%	1.15%
Arcobacter halophilus strain CCUG 53805	0.08%	NP	0.08%	0.08%	0.22%	0.10%
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	0%	NP	NP	0%	0%	1.16%
Mycobacterium tuberculosis KZN 4207	0%	NP	NP	0%	0%	0.08%
Escherichia coli O103:H2 str. 12009	0%	NP	NP	0%	0%	1.43%
Arcobacter halophilus strain CCUG 53805	0%	NP	NP	0%	0%	1.13%

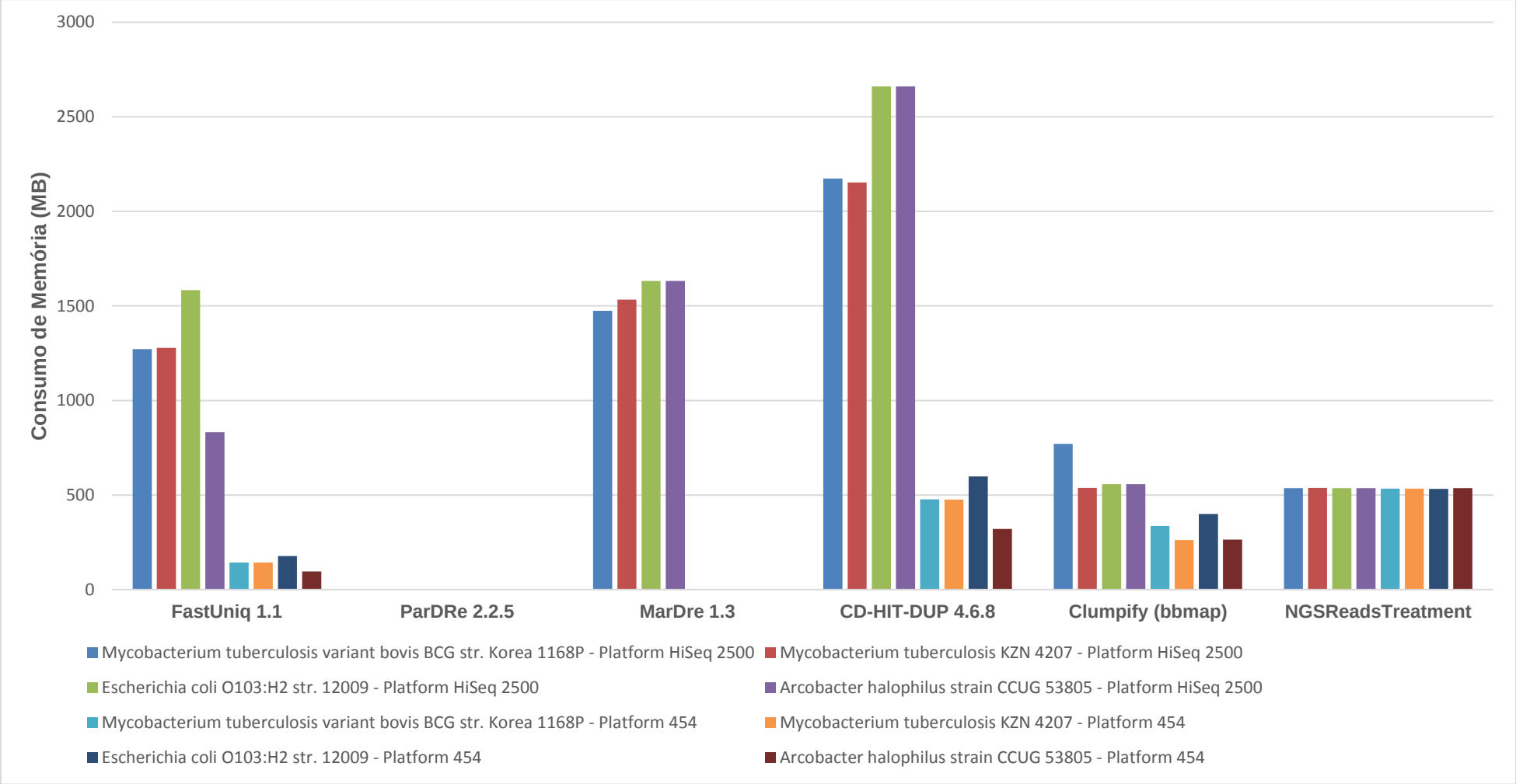
Tabela 7 – Quantidade de memória utilizada por ferramenta em MB para cada conjunto de dados simulados. NP - Não Processado devido a erros.

Organismo	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	1272	NP	1474	2173	771	537
Mycobacterium tuberculosis KZN 4207	1278	NP	1533	2153	538	538
Escherichia coli O103:H2 str. 12009	1583	NP	1632	2660	558	537

Arcobacter halophilus strain CCUG 53805	832	NP	1632	2660	558	537
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P	143	NP	NP	477	337	534
Mycobacterium tuberculosis KZN 4207	143	NP	NP	476	262	534
Escherichia coli O103:H2 str. 12009	177	NP	NP	598	400	533
Arcobacter halophilus strain CCUG 53805	96	NP	NP	321	264	536

Nesses testes, a ferramenta ParDRe foi a única que não processou nenhum *dataset* e a ferramenta MarDRe não processou os dados simulados da plataforma 454. O Clumpify (bbmap) apresentou erro de “Out of memory” no processamento de dois organismos e o CH-HIT-DUP travou o computador nos organismos que tiveram cobertura igual a 300x. Todos os detalhes sobre estes erros e os resultados de processamento por organismo estão disponíveis no Anexo A deste trabalho. A Figura 23 mostra o consumo de memória das ferramentas avaliadas para processar esses dados simulados.

Figura 23 – Quantidade de uso de memória por cada ferramenta computacional para processamento de organismos simulados.



Fonte: Elaborada pelo autor

Na segunda etapa de validação com dados simulados, foram gerados nove *datasets* dos organismos *Escherichia coli O103:H2 str. 12009, Mycobacterium tuberculosis KZN 4207 e Mycobacterium bovis BCG str. Korea 1168P* com coberturas de 100x, 200x e 300x para cada organismo. Todos estes nove conjuntos de dados foram processados por todas as ferramentas para remoção de redundância, exceto a ferramenta ParDRe que não processou nenhum *dataset*, onde o uso de memória de cada ferramenta foi avaliado. Após esse processamento, as leituras únicas de cada um dos conjuntos de dados foram contadas. Essa contagem procura identificar se o número de leituras únicas em um conjunto de dados processado é igual ao número de leituras únicas do conjunto de dados brutos, garantindo assim que apenas leituras redundantes foram removidas na análise.

Como pode ser visto na Tabela 8, o NGSReadsTreatment e as demais ferramentas avaliadas, com exceção da ferramenta Clumpify (bbmap), apresentaram o mesmo número de leituras únicas após o processamento dos dados brutos, o que garante que estas ferramentas removeram apenas as leituras duplicadas destes conjuntos de dados.

A ferramenta Clumpify (bbmap) foi a única que apresentou um número diferente de leituras únicas em relação aos dados brutos, indicando que essa ferramenta pode estar removendo mais dados do que apenas leituras redundantes.

Tabela 8 – Resultado de remoção de redundância por ferramenta para cada organismo simulado com variação de cobertura. NP – Não Processado devido a erros.

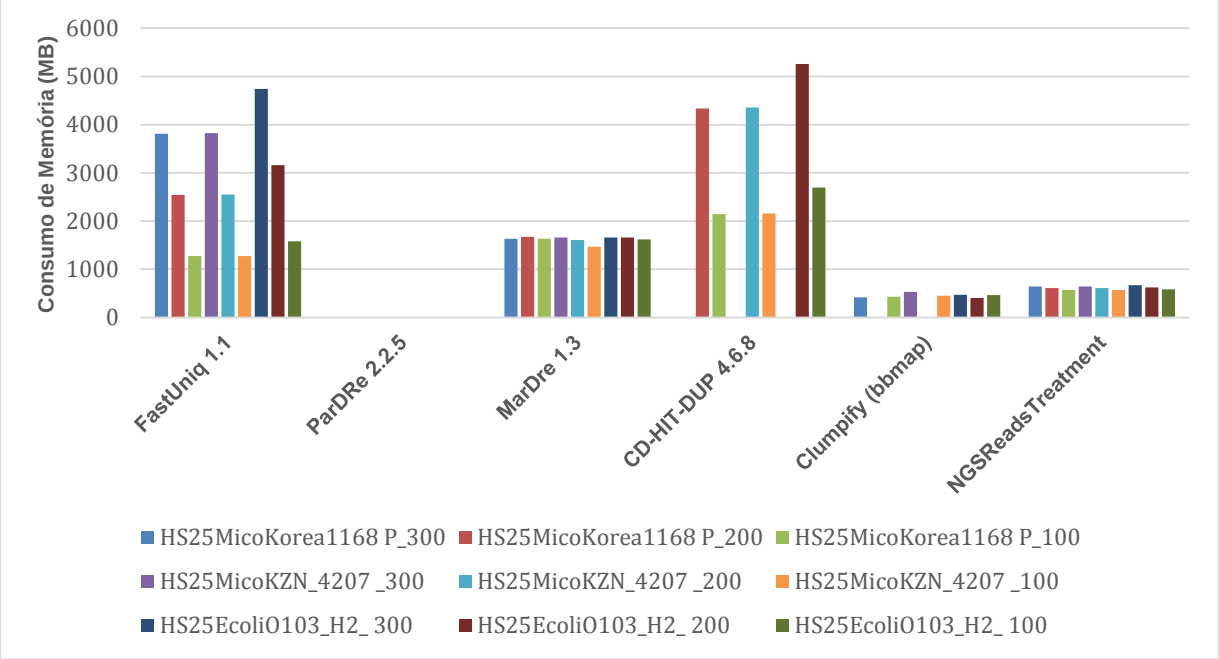
				FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
Organismo	**Cobertura**	**Dataset Bruto**	**Leituras únicas**	**Leituras únicas**	**Leituras únicas**	**Leituras únicas**	**Leituras únicas**	**Leituras únicas**	**Leituras únicas**
Mycobacterium bovis BCG str. Korea 1168P	300x	8753400	8733052	8733052	NP	8733052	NP	8699550	8733052
Mycobacterium bovis BCG str. Korea 1168P	200x	5835600	5826662	5826662	NP	5826662	5826662	NP	5826662
Mycobacterium bovis BCG str. Korea 1168P	100x	2917800	2915548	2915548	NP	2915548	2915548	2911790	2915548
Mycobacterium tuberculosis KZN 4207	300x	8789700	8769474	8769474	NP	8769474	NP	8735712	8769474
Mycobacterium tuberculosis KZN 4207	200x	5859800	5850850	5850850	NP	5850850	5850850	NP	5850850
Mycobacterium tuberculosis KZN 4207	100x	2929900	2927610	2927610	NP	2927610	2927610	2923848	2927610
Escherichia coli O103:H2 str. 12009	300x	10898400	10871060	10871060	NP	10871060	NP	10824528	10871060
Escherichia coli O103:H2 str. 12009	200x	7265600	7253374	7253374	NP	7253374	7253374	7232320	7253374
Escherichia coli O103:H2 str. 12009	100x	3632800	3629674	3629674	NP	3629674	3629674	3624414	3629674

Como não houve diferença na quantidade de leituras redundantes removidas entre o NGSReadsTreatment e as outras ferramentas dessa análise, com exceção da ferramenta Clumpify (bbmap), podemos validar que todos estão removendo apenas dados redundantes dos conjuntos de dados. No entanto, é possível observar na Tabela 9 a disparidade na quantidade de memória necessária para o processamento de dados entre o NGSReadsTreatment e as outras ferramentas, onde o NGSReadsTreatment utilizou uma quantidade inferior de memória para processar a mesma quantidade de dados. A Figura 24 ilustra o consumo de memória desse processamento.

Tabela 9 – Quantidade de memória utilizada por ferramenta em MB para cada conjunto de dados simulados com variação de cobertura. NP - Não Processado devido a erros.

Organismo	Cobertura	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReads Treatment
Mycobacterium bovis BCG str. Korea 1168P	300x	3810	NP	1636	NP	416	647
Mycobacterium bovis BCG str. Korea 1168P	200x	2541	NP	1674	4333	NP	609
Mycobacterium bovis BCG str. Korea 1168P	100x	1273	NP	1636	2148	435	571
Mycobacterium tuberculosis KZN 4207	300x	3826	NP	1658	NP	533	647
Mycobacterium tuberculosis KZN 4207	200x	2552	NP	1607	4353	NP	610
Mycobacterium tuberculosis KZN 4207	100x	1278	NP	1465	2155	451	573
Escherichia coli O103:H2 str. 12009	300x	4743	NP	1658	NP	469	671
Escherichia coli O103:H2 str. 12009	200x	3163	NP	1660	5260	405	625
Escherichia coli O103:H2 str. 12009	100x	1583	NP	1619	2695	462	583

Figura 24 – Quantidade de uso de memória por cada ferramenta computacional para processamento de organismos simulados com variação de cobertura



Fonte: Elaborada pelo autor

7 CONSIDERAÇÕES FINAIS

As análises dos resultados obtidos permitiram verificar a eficiência da estrutura de dados probabilística do *Cuckoo Filter* adotada, pois se mostrou eficaz na remoção de redundâncias de leituras dos arquivos brutos, além de mostrar o uso ideal de memória para o processamento de tarefas com grande volume de dados.

O NGSReadsTreatment remove leituras duplicadas de conjunto de dados fragmentos ou pareados gerados por qualquer plataforma de sequenciamento. Possui uma versão com interface gráfica desenvovlida para facilitar o uso da ferramenta e outra sem interface gráfica que pode ser integrada com outros programas de bioinformática. O NGSReadsTreatment pode ser utilizado tanto em sistemas operacionais UNIX/Linux quanto em sistemas Windows.

Em sua validação o NGSReadsTreatment apresentou o mesmo comportamento na análise de dados reais e simulados. Os resultados simulados do conjunto de dados mostram a eficiência do NGSReadsTreatment na remoção das redundâncias de leituras, conforme listado na Tabela 8 e o baixo consumo de memória conforme listado na Tabela 9.

Os resultados demonstram que o NGSReadsTreatment é uma ferramenta eficiente na remoção de redundância das leituras de sequenciadores NGS e que demanda pouco recurso computacional para processar grande volume de dados. Portanto, é uma alternativa na execução desta tarefa, mesmo que o usuário não possua recursos computacionais elevados.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, E. L. de. **Sequenciamento, montagem e anotação do genoma de *Streptococcus Agalactiae* GBS85147:** uma abordagem comparativa. Dissertação (Mestrado em Bioinformática) – Instituto de Ciências Biológicas, Universidade Federal de Minas Gerais, Belo Horizonte, p. 63. 2015.

AL-HISNAWI, M. & AHMADI, M. **Deep packet inspection using Cuckoo filter**. Annual Conference on New Trends in Information & Communications Technology Applications (NTICT), Baghdad, 2017, pp. 197-202. DOI: 10.1109/NTICT.2017.797611.1.

BLOOM, B. **Space/time trade-offs in hash coding with allowable errors**. Communications of the ACM 13, 422-426 (1970).

BURRIESCI, M.; LEHNERT, E. & PRINGLE, J. **Fulcrum:** condensing redundant reads from high-throughput sequencing studies. Bioinformatics 28, 1324–1327 (2012).

BUSHNELL, Brian. **Introducing Clumpify:** Create 30% Smaller, Faster Gzipped Fastq Files. And remove duplicates. Biostars (2016).

DIJK, Erwin L. van; HÉLÈNE, Auger; YAN, Jaszczyszyn & THERMES, Claude. **Ten years of next-generation sequencing technology.** Trends in Genetics September 2014, Vol. 30, No. 9.

DOHM, J.C.; LOTTAZ, C.; BORODINA, T.& HIMMELBAUER, H. **Substantial biases in ultra-short read data sets from high-throughput DNA sequencing.** Nucleic Acids Res. 2008 Sep; 36 (16).

EBBERT, M. *et al*. **Evaluating the necessity of PCR duplicate removal from next-generation sequencing data and a comparison of approaches.** BMC Bioinformatics 17 (2016).

EL-METWALLY, S.; HAMZA, T.; ZAKARIA, M. & HELMY, M. **Next-Generation Sequence Assembly:** Four Stages of Data Processing and Computational Challenges. *PLoSComputBiol 9 (12): e1003345*, 2013.

EXPÓSITO, R.; VEIGA, J.; GONZÁLES-DOMÍNGUEZ, J. & TOURIÑO, J. **MarDRe:** efficient MapReduce-based removal of duplicate DNA reads in the cloud. Bioinformatics 33, 2762–2764 (2017).

FAN, B.; ANDERSEN, D.; KAMINSKY, M. & MITZENMACHER, M. **Cuckoo Filter.** Proceedings of the 10th ACM International on Conference on emerging Networking

Experiments and Technologies - CoNEXT'14 (2014). doi:10.1145/2674005.2674994.

GILLES *et al*. **Accuracy and quality assessment of 454 GS-FLX Titanium pyrosequencing**. BMC Genomics 2011, 12:245.

GOECKS, J.; NEKRUTENKO, A. & TAYLOR, J. **Galaxy Team.** Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biologics 11, R86, 2010.

GONZÁLEZ-DOMÍNGUES, J. & SCHMIDT, B. **ParDRe:** faster parallel duplicated reads removal tool for sequencing studies: Table 1. Bioinformatics 32, 1562–1564 (2016).

GOODWIN, S.; McPHERSON, J.D. & McCOMBIE, W.R. **Coming of age:** ten years of next-generation sequencing technologies. Nat. Rev. Genet.17, 333–351 (2016).

GOODRICH, M. T.& TAMASSIA, R. **Estruturas de Dados e Algoritmos em Java.** Bookman; Edição: 5 (2013).

GRASHÖFER, J.; JACOB, F. & HARTENSTEIN, H. **Towards Application of Cuckoo Filters in Network Security Monitoring.** 2018 14th International Conference on Network and Service Management (CNSM), Rome, 2018, pp. 373-377.

GRIFFITHS, Anthony J. F.; LEWONTIN, Richard C.; CARROLL, Sean B. & WESSLER, Suzan R. **Introdução à Genética.** Guanabara Koogan; Edição: 11ª (2016).

GUZVIC, Miodrag. **The history of DNA sequencing**. Journal Med Biochem 2013; 32 (4).

HEATHER, James M. & CHAIN, Benjamin. **The sequence of sequencers:** The history of sequencing DNA. Genomics 107 (2016) 1–8.

HENSON, J. & TISCHLER, G. **Ning Z:** Next-generation sequencing and large genome assemblies. Pharmacogenomics 13*: 901-915*, 2012.

HEUSER, Carlos Alberto. **Projeto de Banco de Dados.** Bookman; Edição: 6 (2009).

HUANG, W.; LI, L., MYERS, J.R. & MARTH, G.T. **ART:** a next-generation sequencing read simulator, Bioinformatics, 2012, vol. 28 (pg. 593-594).

KONHEIM, Alan G. **Hashing in Computer Science:** Fifty Years of Slicing and Dicing. Wiley-Interscience; 1 Edition (2010).

LI, W. & GODZIK, A. **Cd-hit:** a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22, 1658–1659 (2006).

MANCONI, A. *et al*. **Removing duplicate reads using graphics processing units.** BMC Bioinformatics 17 (2016).

METZKER, Michael L. **Sequencing technologies –** the next generation. Nature Reviews, Genetics. January 2010. Volume 11.

PAGH, R. & RODLER, F. **Cuckoo hashing.** Journal of Algorithms 51, 122–144 (2004).

PIERCE, Benjamin A. **Genética –** Um Enfoque Conceitual. Guanabara Koogan; Edição: 5ª (2016).

REUTER, Jason A.; SPACEK, Damek V & SNYDER, Michael P. **High-Throughput Sequencing Technologies**. Molecular Cell 58, May 21, 2015.

SANGER, Frederick. *et al*. **DNA sequencing with chain-terminating inhibitors.** Proc Natl Acad Sci U S A. 1977 Dec; 74(12): 5463–5467.

SNUSTAD, D. Peter & SIMMONS, Michael J. **Fundamentos de Genética.** Guanabara Koogan; Edição: 7ª (2017).

TISDALLl, James. **Beginning Perl for Bioinformatics:** An Introduction to Perl for Biologists. O'Reilly Media; February 2009.

XU, H. *et al*. **FastUniq:** A Fast De Novo Duplicates Removal Tool for Paired Short Reads. *PLoS ONE***7**, e52249 (2012).

ZHOU, X. & ROKAS, A. **Prevention, diagnosis and treatment of high-throughput sequencing data pathologies.** Mol Ecol 2014;23(7):1679–700.

ANEXO A – MATERIAL SUPLEMENTAR

RESULTADOS DADOS REAIS

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli RR1	SRR2014554	8192		24.248.885		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Computador travou após 40 min de processamento.					
ParDRe 2.2.5	Computador travou após 17 min de processamento.					
MarDre 1.3	Error: Input files must be the same size in paired-end mode.					
CD-HIT-DUP 4.6.8	Após 01:03:53, Command exited with non-zero status 1. Error: the pair end files contain different number of reads!					
Clumpify (bbmap)	Após 00:07:25, Command exited with non-zero status 1. There appear to be different numbers of reads in the paired input files.					
NGSReadsTreatment	00:20:13	526.63	746.84	104	549	24.179.406

Biblioteca	
Sequenciador	Illumina HiSeq 2000
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli 042	ERR007646	2406		28.221.392		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:24	3.66	30.12	40	3987	13.956.288
ParDRe 2.2.5	00:33:49	19.37	70.63	4	5387	13.956.364
MarDre 1.3	00:03:36	9.72	117.67	58	1653	13.956.288
CD-HIT-DUP 4.6.8	00:25:24	40.35	91.32	8	5393	13.956.288
Clumpify (bbmap)	00:03:25	11.22	103.08	55	870	13.927.100
NGSReadsTreatment	00:11:22	302.21	413.13	107	543	13.970.954

Biblioteca	
Sequenciador	Illumina Genome Analyzer
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli P12b	SRR2000272	1351		2.991.478		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:25	1.53	8.29	38	1722	2.967.034
ParDRe 2.2.5	00:01:07	2.29	34.65	54	2278	2.967.266
MarDre 1.3	ERROR: Input files must be the same size in Pareada-end mode.					
CD-HIT-DUP 4.6.8	00:01:12	3.37	24.72	38	3076	2.967.266
Clumpify (bbmap)	00:01:02	4.55	53.58	92	423	2.963.080
NGSReadsTreatment	00:02:24	58.57	91.27	103	537	2.934.404

Biblioteca	
Sequenciador	Illumina MiSeq
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli KLY	SRR1424625	1682		6.866.668		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:34	2.03	15.22	50	2571	6.866.668
ParDRe 2.2.5	00:01:14	3.22	41.39	59	3586	6.866.668
MarDre 1.3	00:02:34	8.29	80.50	57	1676	6.866.668
CD-HIT-DUP 4.6.8	00:01:21	4.12	33.49	56	4097	6.866.668
Clumpify (bbmap)	00:01:57	7.63	79.14	73	455	6.852.786
NGSReadsTreatment	00:05:26	138.91	206.63	105	539	6.802.266

Biblioteca	
Sequenciador	Illumina HiSeq 2000
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli O25b:H4-ST131	SRR933487	1070		3.214.312		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:14	1.23	7.11	58	1449	3.191.014
ParDRe 2.2.5	00:00:24	1.77	21.87	95	1950	3.191.016
MarDre 1.3	00:00:55	4.86	44.13	87	1411	3.191.014
CD-HIT-DUP 4.6.8	00:00:22	1.97	14.99	76	2063	3.191.014
Clumpify (bbmap)	00:00:24	3.34	29.15	135	2215	3.178.716
NGSReadsTreatment	00:02:30	65.33	94.00	105	538	3.153.156

Biblioteca	
Sequenciador	Illumina Genome Analyzer IIx
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Kineococcus rhizosphaerae DSM 19711	SRR6479489	2048		5.641.334		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:50	2.39	14.47	33	2629	5.633.496
ParDRe 2.2.5	00:01:44	4.77	54.83	56	3629	5.633.504
MarDre 1.3	00:03:06	10.64	81.50	49	1652	5.633.496
CD-HIT-DUP 4.6.8	00:01:38	4.65	31.95	37	4313	5.633.496
Clumpify (bbmap)	00:02:36	9.84	74.34	53	3454	5.630.770
NGSReadsTreatment	00:04:37	110.41	173.28	102	538	5.572.908

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Kineococcus xinjiangensis DSM 22857	SRR6479482	2168		5.971.022		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:43	4.45	19.19	22	2783	5.962.418
ParDRe 2.2.5	00:05:23	6.00	61.11	20	3850	5.962.424
MarDre 1.3	00:03:13	10.82	88.12	51	1647	5.962.418
CD-HIT-DUP 4.6.8	00:01.36	5.18	32.34	38	4501	5.962.418
Clumpify (bbmap)	00:02:52	10.45	79.49	52	3616	5.960.172
NGSReadsTreatment	00:05:01	122.05	191.75	104	538	5.900.978

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium tuberculosis F11	SRR974839	1936		14.110.696		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:21	3.62	18.26	26	2725	7.233.122
ParDRe 2.2.5	00:09:39	4.87	534.85	93	3825	7.233.132
MarDre 1.3	00:10:47	11.39	462.76	73	1634	7.233.122
CD-HIT-DUP 4.6.8	00:01:44	5.43	36.38	40	4499	7.233.122
Clumpify (bbmap)	00:02:39	10.80	73.15	52	2625	7.186.020
NGSReadsTreatment	00:05:48	148.79	224.59	106	540	7.185.138

Biblioteca	
Sequenciador	Illumina HiSeq 2000
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium tuberculosis XDR KZN 4207	SRR1144800	1884		7.033.428		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:20	2.98	16.53	24	2633	7.028.960
ParDRe 2.2.5	00:01:32	3.23	46.44	53	3730	7.028.960
MarDre 1.3	00:03:45	10.62	85.10	42	1653	7.028.960
CD-HIT-DUP 4.6.8	00:02:59	5.16	35.54	22	4351	7.028.960
Clumpify (bbmap)	00:03:26	10.19	68.26	38	3250	7.026.512
NGSReadsTreatment	00:05:30	140.15	209.60	105	540	6.967.590

Biblioteca	
Sequenciador	Illumina HiSeq 2000
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Arcobacter halophilus strain CCUG 53805	SRR7587111	588		670.813		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in open left fastq file @SRR7587111.1 FZNUX9M02GJ4ET length=327 for read!					
ParDRe 2.2.5	00:00:23	0.91	15.77	72	888	668.356
MarDre 1.3	00:00:34	3.23	28.22	92	1118	668.356
CD-HIT-DUP 4.6.8	00:00:14	1.14	12.22	95	1561	668.356
Clumpify (bbmap)	00:00:13	1.93	24.38	192	769	667.959
NGSReadsTreatment	00:38:00	14.65	25.12	102	533	664.953

Biblioteca	
Sequenciador	454 GS FLX+
Tipo de Leitura	Pareada

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Rhodopirellula baltica strain:BR-MGV	SRR7819959	1984		3.207.713		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in open left fastq file @SRR7819959.1 1 length=209 for read!					
ParDRe 2.2.5	00:01:30	3.39	62.44	72	3197	3.183.983
MarDre 1.3	00:02:42	10.02	87.13	59	1665	3.183.983
CD-HIT-DUP 4.6.8	cd-hit-dup: cdhit-dup.cxx:193: int HashingDepth(int, int): Assertion `len >= min' failed.					
Clumpify (bbmap)	00:02:17	6.31	90.72	70	659	3.181.912
NGSReadsTreatment	00:03:12	64.13	107.75	89	537	3.164.133

Biblioteca	
Sequenciador	Ion Torrent PGM
Tipo de Leitura	Fragmentos

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Surveillance of Escherichia coli O157:H7 in Romania	ERR2375157	1201		2.106.268		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in open left fastq file @ERR2375157.1 1 length=294 for read!					
ParDRe 2.2.5	00:00:48	2.03	32.19	70	1989	2.104.786
MarDre 1.3	00:01:32	6.23	53.73	64	1394	2.104.786
CD-HIT-DUP 4.6.8	cd-hit-dup: cdhit-dup.cxx:193: int HashingDepth(int, int): Assertion `len >= min' failed.					
Clumpify (bbmap)	00:00:51	4.13	48.17	101	744	2.104.540
NGSReadsTreatment	00:02:05	42.26	70.90	89	537	2.073.554

Biblioteca	
Sequenciador	Ion Torrent PGM
Tipo de Leitura	Fragmentos

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Rathayibacter tritici strain:FH211, CT102, ATCC 11403	SRR6799098	157		160.403		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in open left fastq file @SRR6799098.1 HTUX4QU01ANJ8O length=78 for read!					
ParDRe 2.2.5	00:00:06	0.25	4.16	70	247	157.022
MarDre 1.3	00:00:16	1.50	15.15	102	913	157.022
CD-HIT-DUP 4.6.8	00:00:03	0.27	3.33	99	481	157.022
Clumpify (bbmap)	00:00:02	0.30	5.31	208	373	156.842
NGSReadsTreatment	00:00:08	3.45	6.63	117	531	156.835

Biblioteca	
Sequenciador	454 GS Junior
Tipo de Leitura	Fragmentos

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Salmonella enterica subsp. enterica serovar Fresno strain:USMARC69835	SRR7905974	2990		163.468		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in read from FASTQ file list!					
ParDRe 2.2.5	00:02:08	4.60	83.50	68	3796	163.464
MarDre 1.3	java.lang.Exception: java.lang.StringIndexOutOfBoundsException: String index out of range: 19					
CD-HIT-DUP 4.6.8	Computador travou após16 min de processamento.					
Clumpify (bbmap)	00:03:50	7.16	120.72	55	961	163.464
NGSReadsTreatment	00:02:08	8.20	33.51	32	526	163.249

Biblioteca	
Sequenciador	PacBio RS II
Tipo de Leitura	Fragmentos

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Staphylococcus aureus Cystic Fibrosis Isolates	SRR7739756	1336		86.389		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in read from FASTQ file list!					
ParDRe 2.2.5	00:00:57	2.34	35.76	66	1704	86.386
MarDre 1.3	INFO mapreduce.Job: Job job_local1848694102_0001 failed with state FAILED due to: NA.					
CD-HIT-DUP 4.6.8	cd-hit-dup: cdhit-dup.cxx:193: int HashingDepth(int, int): Assertion `len >= min' failed.					
Clumpify (bbmap)	00:01:08	3.44	52.06	80	700	86.386
NGSReadsTreatment	00:00:17	3.35	13.64	95	532	86.324

Biblioteca	
Sequenciador	MinION
Tipo de Leitura	Fragmentos

Organismo	Biblioteca	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Pseudomonas aeruginosa microevolution during chronic infection of the CF lung	ERR2162181	246		1.146.696		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	Error in open left fastq file @ERR2162181.1 1 length=75 for read.					
ParDRe 2.2.5	00:00:09	0.43	6.78	73	625	1.146.694
MarDre 1.3	00:00:29	1.93	22.90	83	947	1.146.694
CD-HIT-DUP 4.6.8	00:00:04	0.58	4.34	99	779	1.146.694
Clumpify (bbmap)	Exception in thread "main" java.lang.AssertionError: ASCII encoding for quality (currently ASCII-33) appears to be wrong for input quality 15 for base T at lines 1 and 3, position 67.					
NGSReadsTreatment	00:00:53	22.87	35.42	108	535	1.142.899

Biblioteca	
Sequenciador	AB 5500xl Genetic Analyzer
Tipo de Leitura	Fragmentos

RESULTADO DADOS SIMULADOS

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P		910MB		2.917.800		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:13	1.14	6.52	56	1272	2.915.606
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:00:57	5.35	45.13	88	1474	2.915.606
CD-HIT-DUP 4.6.8	00:00:43	2.28	14.08	37	2173	2.915.606
Clumpify (bbmap)	00:00:34	2.88	36.74	113	771	2.911.848
NGSReadsTreatment	00:02:19	58.79	90.63	107	537	2.884.114

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Mycobacterium tuberculosis KZN 4207		914MB		2.929.900		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:12	1.15	6.57	60	1278	2.927.650
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:01:28	5.76	45.26	57	1533	2.927.650
CD-HIT-DUP 4.6.8	00:00:35	2.09	13.99	45	2153	2.927.650
Clumpify (bbmap)	00:00:28	2.74	39.03	148	538	2.923.922
NGSReadsTreatment	00:02:19	58.38	90.80	107	538	2.895.866

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Escherichia coli O103:H2 str. 12009		1132MB		3.632.800		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:17	1.42	8.08	53	1583	3.629.678
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:02:22	7.66	54.62	43	1632	3.629.678
CD-HIT-DUP 4.6.8	00:00:56	2.76	18.64	38	2660	3.629.678
Clumpify (bbmap)	00:00:46	3.82	46.82	108	558	3.624.308
NGSReadsTreatment	00:02:54	72.73	111.84	105	537	3.580.676

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Arcobacter halophilus strain CCUG 53805		588MB		1.875.000		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:05	0.74	4.29	89	832	1.873.432
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:00:51	4.15	32.80	72	1250	1.873.432
CD-HIT-DUP 4.6.8	00:00:12	1.12	8.19	76	1363	1.873.432
Clumpify (bbmap)	00:00:13	2.23	24.07	201	569	1.870.872
NGSReadsTreatment	00:01:27	35.45	59.45	108	536	1.860.626

Biblioteca	
Sequenciador	Illumina HiSeq 2500
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P		104MB		251.688		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:00.63	0.16	0.46	99	143	251.682
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	Error: Input files must be the same size in Pareada-end mode.					
CD-HIT-DUP 4.6.8	00:00:02	0.26	2.15	98	477	251.688
Clumpify (bbmap)	00:00:03	0.30	3.91	137	337	251.688
NGSReadsTreatment	00:00:17	8.13	11.29	109	534	251.436

Biblioteca	
Sequenciador	454
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Mycobacterium tuberculosis KZN 4207		104MB		252.954		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:00.62	0.14	0.48	100	143	252.954
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	Error: Input files must be the same size in Pareada-end mode.					
CD-HIT-DUP 4.6.8	00:00:03	0.32	2.23	68	476	252.954
Clumpify (bbmap)	00:00:01	0.25	3.75	300	262	252.954
NGSReadsTreatment	00:00:12	5.23	8.84	115	534	252.748

Biblioteca	
Sequenciador	454
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Escherichia coli O103:H2 str. 12009		130MB		313.944		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:00.78	0.14	0.63	100	177	313.938
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	Error: Input files must be the same size in Pareada-end mode.					
CD-HIT-DUP 4.6.8	00:00:04	0.46	2.66	65	598	313.944
Clumpify (bbmap)	00:00:01	0.37	4.18	265	400	313.944
NGSReadsTreatment	00:00:14	6.25	10.53	115	533	313.542

Biblioteca	
Sequenciador	454
Tipo de Leitura	Pareada

Organismo	**Dado Simulado**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Arcobacter halophilus strain CCUG 53805		70MB		161.900		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:00.42	0.07	0.32	93	96	161.900
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	Error: Input files must be the same size in Pareada-end mode.					
CD-HIT-DUP 4.6.8	0:00:02	0.17	1.47	65	321	161.900
Clumpify (bbmap)	00:00:01	0.21	3.03	166	264	161.900
NGSReadsTreatment	0:00:08	3.30	6.20	118	536	161.812

Biblioteca	
Sequenciador	454
Tipo de Leitura	Pareada

RESULTADOS DADOS SIMULADOS COM DIFERENTES VALORES DE COBERTURA

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli O103:H2 str. 12009	100x	1132MB		3.632.800		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:20	1.47	8.51	47	1583	3.629.674
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:01:19	6.69	53.29	75	1619	3.629.674
CD-HIT-DUP 4.6.8	00:00:49	2.78	18.46	43	2695	3.629.674
Clumpify (bbmap)	00:01:06	4.76	54.45	88	462	3.624.414
NGSReadsTreatment	00:02:54	71.49	114.19	106	583	3.629.674

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium bovis BCG str. Korea 1168P	100x	908MB		2.917.800		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:14	1.08	6.60	52	1273	2.915.548
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:01:30	5.60	44.85	55	1636	2.915.548
CD-HIT-DUP 4.6.8	00:00:39	2.16	14.17	41	2148	2.915.548
Clumpify (bbmap)	00:00:54	3.15	38.83	76	435	2.911.790
NGSReadsTreatment	00:02:30	58.81	93.24	101	571	2.915.548

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium tuberculosis KZN 4207	100x	914MB		2.929.900		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:00:24	1.42	6.88	33	1278	2.927.610
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:01:20	5.68	45.61	63	1465	2.927.610
CD-HIT-DUP 4.6.8	00:00:35	2.07	14.04	45	2155	2.927.610
Clumpify (bbmap)	00:00:30	2.65	39.96	139	451	2.923.848
NGSReadsTreatment	00:02:17	56.57	90.82	106	573	2.927.610

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	**Cobertura**	**Tamanho do Arquivo em MB**		**Total de leituras por Dataset**		
Escherichia coli O103:H2 str. 12009	200x	2266MB		7.265.600		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:33	3.79	18.55	23	3163	7.253.374
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:06:58	16.93	98.24	27	1660	7.253.374
CD-HIT-DUP 4.6.8	00:04:59	6.35	39.81	15	5260	7.253.374
Clumpify (bbmap)	00:03:28	9.95	118.71	61	405	7.232.320
NGSReadsTreatment	00:06:11	147.42	224.89	100	625	7.253.374

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium bovis BCG str. Korea 1168P	200x	1814MB		5.835.600		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:11	2.72	14.89	24	2541	5.826.662
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:04:09	11.59	82.91	37	1674	5.826.662
CD-HIT-DUP 4.6.8	00:01:25	4.36	29.67	40	4333	5.826.662
Clumpify (bbmap)	Erro de falta de memória.					
NGSReadsTreatment	00:05:05	121.96	188.88	101	609	5.826.662

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium tuberculosis KZN 4207	200x	1828MB		5.859.800		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:11	3.10	15.38	25	2552	5.850.850
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:04:44	11.32	80.51	32	1607	5.850.850
CD-HIT-DUP 4.6.8	00:01:27	4.80	30.16	40	4353	5.850.850
Clumpify (bbmap)	Erro de falta de memória.					
NGSReadsTreatment	00:05:02	119.56	188.09	101	610	5.850.850

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Escherichia coli O103:H2 str. 12009	300x	3400MB		10.898.400		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:02:34	6.21	31.04	24	4743	10.871.060
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:09:22	26.03	144.80	30	1658	10.871.060
CD-HIT-DUP 4.6.8	Computador travou após 58 min de processamento.					
Clumpify (bbmap)	00:06:10	15.65	180.36	52	469	10.824.528
NGSReadsTreatment	00:09:38	232.77	357.18	101	671	10.871.060

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium bovis BCG str. Korea 1168P	300x	2722MB		8.753.400		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:01:19	3.70	22.74	33	3810	8.733.052
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:06:26	17.77	114.96	34	1636	8.733.052
CD-HIT-DUP 4.6.8	Computador travou após 15 min de processamento.					
Clumpify (bbmap)	00:03:39	10.95	134.95	66	416	8.699.550
NGSReadsTreatment	00:07:31	178.79	271.85	99	647	8.733.052

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

Organismo	Cobertura	Tamanho do Arquivo em MB		Total de leituras por Dataset		
Mycobacterium tuberculosis KZN 4207	300x	2742MB		8.789.700		
Ferramenta	**Tempo de Processamento (em horas: minutos: segundos)**	**Número total (segundos) de CPU no modo kernel**	**Número total (segundos) de CPU no modo usuário**	**Percentual de Uso de CPU**	**Quantidade de memória utilizada em MB**	**Total de leituras após tratamento**
FastUniq 1.1	00:03:24	6.53	30.32	18	3826	8.769.474
ParDRe 2.2.5	O processamento foi interrompido pela ferramenta.					
MarDre 1.3	00:06:08	17.83	112.64	35	1658	8.769.474
CD-HIT-DUP 4.6.8	Computador travou após 17 min de processamento.					
Clumpify (bbmap)	00:03:55	12.74	137.06	63	533	8.735.712
NGSReadsTreatment	00:07:45	185.19	278.90	99	647	8.769.474

Biblioteca	
Sequenciador	HiSeq2500
Tipo de Leitura	Pareada

ANEXO B – PRODUÇÃO CIENTÍFICA

Artigo publicado na Revista ***Nature Scientific Reports:*** NGSReadsTreatment – A Cuckoo Filter-based Tool for Removing Duplicate Reads in NGS Data

SCIENTIFIC REPORTS
nature research

OPEN

NGSReadsTreatment – A Cuckoo Filter-based Tool for Removing Duplicate Reads in NGS Data

Received: 16 October 2018
Accepted: 1 August 2019
Published online: 12 August 2019

Antonio Sérgio Cruz Gaia[1], Pablo Henrique Caracciolo Gomes de Sá[2], Mônica Silva de Oliveira[1] & Adonney Allan de Oliveira Veras[1]

The Next-Generation Sequencing (NGS) platforms provide a major approach to obtaining millions of short reads from samples. NGS has been used in a wide range of analyses, such as for determining genome sequences, analyzing evolutionary processes, identifying gene expression and resolving metagenomic analyses. Usually, the quality of NGS data impacts the final study conclusions. Moreover, quality assessment is generally considered the first step in data analyses to ensure the use of only reliable reads for further studies. In NGS platforms, the presence of duplicated reads (redundancy) that are usually introduced during library sequencing is a major issue. These might have a serious impact on research application, as redundancies in reads can lead to difficulties in subsequent analysis (e.g., *de novo* genome assembly). Herein, we present NGSReadsTreatment, a computational tool for the removal of duplicated reads in paired-end or single-end datasets. NGSReadsTreatment can handle reads from any platform with the same or different sequence lengths. Using the probabilistic structure Cuckoo Filter, the redundant reads are identified and removed by comparing the reads with themselves. Thus, no prerequisite is required beyond the set of reads. NGSReadsTreatment was compared with other redundancy removal tools in analyzing different sets of reads. The results demonstrated that NGSReadsTreatment was better than the other tools in both the amount of redundancies removed and the use of computational memory for all analyses performed. Available in https://sourceforge.net/projects/ngsreadstreatment/.

The advent of Next-Generation Sequencing (NGS) technologies in mid-2005 provided significant breakthroughs in the omics fields. These platforms can generate millions of reads in a short time; for instance, Illumina NextSeq is capable of generating 400 million reads per round. The genomic library must be prepared prior to the actual sequencing and one task included in this stage is polymerase chain reaction (PCR) amplification[1].

PCR generates a super-representation of a sample fragment, giving rise to the concept of coverage, where the genetic material of an organism to be sequenced presents a several-fold multiplication of its expected size. This super-representation is important for several analyses, including frameshift curation and single nucleotide polymorphism analyses, among others[1].

However, some analyses are impacted by this super-representation, such as *de novo* assembly and the final scaffolding process. Moreover, the tasks demand a high computational cost, and duplication gives rise to false positives with overlapping contigs, as well as their subsequent extension due to the high number of connections. Consequently, false negatives arise as a result of the overlapping conflicts generated by the duplications[2].Thus, the development of computational methods that can remove sequencing read redundancies is important. Several software solutions have been developed over the years to address this situation. GPU-DupRemoval (by Removing GPU Duplicates) aims to remove duplicate reads using graphical processing units (GPUs) generated with the Illumina platform. The task is divided into two phases: the clustering of possible duplicate sequences according to their prefix, followed by comparison of the sequence suffixes in each cluster to detect and remove redundancies[3].

The FASTX-Toolkit Collapser (http://hannonlab.cshl.edu/fastx_toolkit), FastUniq[4], Fulcrum[5], and CD-HIT[6] tools employ an alignment-free strategy. FASTX-Toolkit Collapser is able to identify and remove identical sequences from single-end reads. FastUniq, on the other hand, is designed to remove identical duplicates in three

[1]Postgraduate Program in Applied Computing, Federal University of Pará (UFPA), Pará, Brazil. [2]Federal Rural University of Amazonia Campus Tomé-Açu (UFRA), Pará, Brazil. Antonio Sérgio Cruz Gaia and Pablo Henrique Caracciolo Gomes de Sá contributed equally. Correspondence and requests for materials should be addressed to A.A.d.O.V. (email: allanveras@ufpa.br)

Organism	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReadsTreatment
SRR2014554	NP	NP	NP	NP	NP	0.29%
ERR007646	50.55%	50.55%	50.55%	50.55%	50.65%	50.50%
SRR2000272	0.82%	0.81%	NP	0.81%	0.95%	1.91%
SRR1424625	0%	0%	0%	0%	0.20%	0.94%
SRR933487	0.72%	0.72%	0.72%	0.72%	1.11%	1.90%
SRR6479489	0.14%	0.14%	0.13%	0.14%	0.19%	1.21%
SRR6479482	0.14%	0.14%	0.14%	0.14%	0.18%	1.17%
SRR974839	48.74%	48.74%	48.74%	48.74%	49.07%	49.08%
SRR1144800	0.06%	0.06%	0.06%	0.06%	0.10%	0.94%
SRR7587111	NP	0.37%	0.37%	0.37%	0.43%	0.87%
SRR7819959	NP	0.74%	0.74%	NP	0.80%	1.36%
ERR2375157	NP	0.07%	0.07%	NP	0.08%	1.55%
SRR6799098	NP	2.11%	2.11%	2.1%	2.22%	2.22%
SRR7905974	NP	0%	NP	NP	0%	0.13%
SRR7739756	NP	0%	NP	NP	0%	0.08%
ERR2162181	NP	0%	0%	0%	NP	0.33%

Table 1. Percentage of read redundancy removal per tool for each organism. NP - not processed owing to errors.

Organism	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReadsTreatment
SRR2014554	NP	NP	NP	NP	NP	549
ERR007646	3987	5387	1653	5393	870	543
SRR2000272	1722	2278	NP	3076	423	537
SRR1424625	2571	3586	1676	4097	455	539
SRR933487	1449	1950	1411	2063	2215	538
SRR6479489	2629	3629	1652	4313	3454	538
SRR6479482	2783	3850	1647	4501	3616	538
SRR974839	2725	3825	1634	4499	2625	540
SRR1144800	2633	3730	1653	4351	3250	540
SRR7587111	NP	888	1118	1561	769	533
SRR7819959	NP	3197	1665	NP	659	537
ERR2375157	NP	1989	1394	NP	744	537
SRR6799098	NP	247	913	481	373	531
SRR7905974	NP	3796	NP	NP	961	526
SRR7739756	NP	1704	NP	NP	700	532
ERR2162181	NP	625	947	779	NP	535

Table 2. Memory amount used by each tool in megabyte. NP - not processed owing to errors.

steps: initially, all paired reads are loaded into the memory; subsequently, the read pairs are sorted, and finally the duplicate sequences are identified by comparing the adjacent read pairs in the sorted list.

Fulcrum is able to identify duplicates that are fully or partially identical. Reads identified as possible duplicates are kept in different files, whose maximum size is defined by the user. The read sequences within each file are compared to identify duplicates[5].

CD-HIT has two different tools for removing duplicates of single-end and paired-end reads generated with the Illumina platform. CD-HIT-454 parses libraries generated with 454 to identify exactly identical duplicates[6].

The majority of the existing tools are designed to serve a particular sequencing platform. Thus, we present the NGSReadsTreatment tool for the removal of read redundancies for any NGS platform, based on the probabilistic structure of Cuckoo Filter.

Results and Discussion

The reads sets of the sixteen organisms (real datasets) were processed using the FastUniq 1.1[4], ParDRe 2.2.5[7], MarDre 1.3[8], CD-HIT-DUP 4.6.86[6], Clumpify (https://sourceforge.net/projects/bbmap), and NGSReadsTreatment computational tools. The percentage of redundancy removal for each organism as well as an evaluation of the total memory used per tool is shown in Tables 1 and 2, respectively.

Table 1 shows that NGSReadsTreatment obtained a greater percentage of redundant read removals for thirteen of the sixteen organisms analyzed, being that in an organism the percentage of removal equal to that of another tool used in the test; that is, it was able to identify and remove the largest amount of redundancies. Some datasets

Organism	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReadsTreatment
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P - Platform HiSeq 2500	0.08%	NP	0.08%	0.08%	0.20%	0.77%
Mycobacterium tuberculosis KZN 4207 - Platform HiSeq 2500	0.08%	NP	0.08%	0.08%	0.20%	0.05%
Escherichia coli O103:H2 str. 12009 - Platform HiSeq 2500	0.09%	NP	0.09%	0.09%	0.23%	1.15%
Arcobacter halophilus strain CCUG 53805 - Platform HiSeq 2500	0.08%	NP	0.08%	0.08%	0.22%	0.10%
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P - Platform 454	0%	NP	NP	0%	0%	1.16%
Mycobacterium tuberculosis KZN 4207 - Platform 454	0%	NP	NP	0%	0%	0.08%
Escherichia coli O103:H2 str. 12009 - Platform 454	0%	NP	NP	0%	0%	1.43%
Arcobacter halophilus strain CCUG 53805 - Platform 454	0%	NP	NP	0%	0%	0.13%

Table 3. Percentage of read redundancy removal per tool for each simulated dataset. NP - not processed owing to errors.

of organisms, for example SRR2000272, SRR7905974 and SRR2014554, experienced processing problems with the other computational tools: computer crashes during execution and processing failure due to the existence of orphan sequences in the read files. The tools that presented 0% were not able to remove any redundancy in the dataset, despite processing the data normally.

For the SRR2014554 organism, only the NGSReadsTreatment was successful in processing the 4-GB dataset. All the other tested tools presented errors during read processing.

Table 2 lists the total memory used by each tool in the processing of the raw reads. Similar to the results described in Table 1, the dataset of some organisms presented problems during the execution by the other tools. However, it was possible to use the NGSReadsTreatment software in all cases, thereby also demonstrating its efficiency in the use of memory, since it was the only tool that used the least computational memory among all the tested tools in most analyses.

The FastUniq software does not support single-end reads in its analyzes, so it was not possible to perform the processing of reads of this type with the tool. However, in all cases it was possible to use the NGSReadsTreatment, also demonstrating its efficiency in processing paired-end and single-end reads, with a reduced computational memory usage.

To improve the validation of NGSReadsTreatment the same analyzes performed with the real datasets (sixteen organisms) were performed with simulated datasets from ART tool[9]. It can be observed that NGSReadsTreatment has proved to be efficient for both redundancy removal and memory usage as shown in the Tables 3 and 4.

Most errors were observed during the processing of the single-end reads, all details on the errors and all processing results per organism are available in the supplementary material.

In the third validation step, after the generation of the nine datasets with different coverage, the reads were counted to determine the amount of reads, number of unique reads and the amount of redundant reads in the raw data of each dataset (last table of the section simulated data with different coverage values in the Supplementary Material).

All nine datasets were processed by all tools for redundancy removal, where the memory usage by each tool was evaluated. After this processing, the unique reads of each of the datasets were counted. This count seeks to identify whether the number of unique reads in a processed dataset (Supplementary Material) is equal to the number of unique reads of the raw dataset, thus ensuring that only redundant reads were removed in the analysis.

As can be seen in Supplementary Material, the NGSReadsTreatment and all the tools used, with the exception of the Clumpify (bbmap) tool, were able to reach the number of unique reads equal to the raw data, thus ensuring that all these tools succeeded in removing only the redundant reads of each dataset.

The Clumpify (bbmap) tool was the only one that presented a different number of unique reads in relation to the raw data, indicating that this tool may be removing more data than just redundant reads.

As there was no difference in the amount of redundant reads removed between NGSReadsTreatment and the other tools of this analysis, with exception of the Clumpify (bbmap) tool, we can validate that all are removing only redundant data from the datasets, however, it is possible to observe the disparity in the amount of memory required for the data processing between NGSReadsTreatment and the other tools, where NGSReadsTreatment used a much smaller amount of memory to process the same amount of data (Fig. 1). All processing results per organism are available in the supplementary material.

The analysis of the results obtained herein allowed verification of the efficiency of the adopted Cuckoo Filter probabilistic data structure, as it proved effective in removing read redundancies from the raw files, besides showing optimal memory usage for task processing. The NGSReadsTreatment tool is capable of handling single-end and paired-end files, and is available in two versions: one with a graphical interface and control of processing status through a database. Thus, in case of some kind of error or if the user wishes to interrupt processing, it can be resumed. A version without a graphical interface is also available.

The NGSReadsTreatment presented the same behavior in the analysis of both real data and simulated data. The simulated dataset results show the efficiency of the NGSReadsTreatment in the removal of the reads redundancies as listed in Table 3.

Thus, it is concluded that NGSReadsTreatment has proven to be an efficient tool in removing redundancy from NGS reads, thus being an alternative in the execution of this task even if the user does not have high computational resources.

Organism	FastUniq 1.1	ParDRe 2.2.5	MarDre 1.3	CD-HIT-DUP 4.6.8	Clumpify (bbmap)	NGSReadsTreatment
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P - Platform HiSeq2500	1272	NP	1474	2173	771	537
Mycobacterium tuberculosis KZN 4207 - Platform HiSeq2500	1278	NP	1533	2153	538	538
Escherichia coli O103:H2 str. 12009 - Platform HiSeq2500	1583	NP	1632	2660	558	537
Arcobacter halophilus strain CCUG 53805 - Platform HiSeq2500	832	NP	1250	1363	569	536
Mycobacterium tuberculosis variant bovis BCG str. Korea 1168P - Platform 454	143	NP	NP	477	337	534
Mycobacterium tuberculosis KZN 4207 - Library 454	143	NP	NP	476	262	534
Escherichia coli O103:H2 str. 12009 - Platform 454	177	NP	NP	598	400	533
Arcobacter halophilus strain CCUG 53805 - Platform 454	96	NP	NP	321	264	536

Table 4. Memory amount used by each tool in megabyte for each simulated dataset. NP - not processed owing to errors.

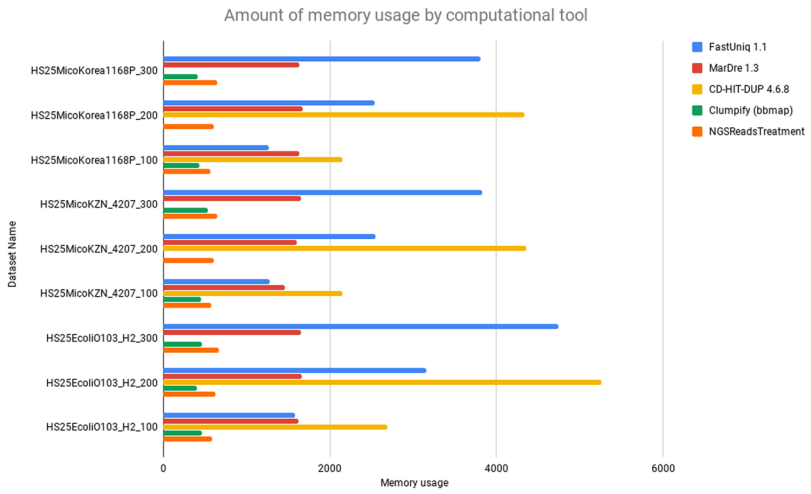


Figure 1. Evaluation of memory usage for each computational tool in the processing of simulated datasets.

Methodology

Programming language and database. NGSReadsTreatment was developed in JAVA language (http://www.oracle.com/) and the Swing library was used to create the graphical interface (http://www.oracle.com/). Maven (https://maven.apache.org/) was used for dependency management and build automation. Its main features include the following (among others): simplified project configuration following best practices, automated dependency management, and JAR generation with all the dependencies used in the project. The project management was performed with SQLite version 3 (https://www.sqlite.org/).

Redundancy removal. Cuckoo Filter[10] was used to remove redundancies from the reads in the raw files. It is a quick and effective probabilistic data structure for cluster association queries. Developed by Fan, Andersen, Kaminsky, and Mitzenmacher, Cuckoo Filter emerged as an enhancement to Bloom Filter[11], introducing support for dynamic item deletion, improved search performance, and improved space efficiency for low false-positive applications.

The Cuckoo Filter uses cuckoo hashing[12] to resolve collisions and basically consists of a compact cuckoo hash table that stores the fingerprints of inserted items. Each fingerprint is a string of bits derived from the hash of the item to be inserted.

A cuckoo hash table consists of a two-dimensional array where the rows correspond to the associative units called buckets and their cells are called slots. A bucket can contain multiple slots and each slot is used to store a single fingerprint of predefined size[10]. For example a cuckoo filter (2,4) has slots that store 2-bit fingerprints and each table bucket can hold up to 4 fingerprints.

In the process of removing redundancy is generated for each read a fingerprint and checked if it is contained in the cuckoo hash table, if the answer is false the fingerprint is inserted into the table and the read is stored in a text file, otherwise the read is discarded.

It is worth mentioning that these probabilistic structures[10] do not provide false negatives, which allows greater efficiency in the removal of duplicate reads from the raw file.

Evaluation of computational cost. Linux's *time* software (http://man7.org/linux/man-pages/man1/time.1.html) was used to generate statistics for a command, shell script, or any executed program. The statistics included the time spent by the program in the user mode, the time spent by the program in the kernel mode,

Organism	SRA Access number	File size by Dataset	Total of Reads by Dataset	Type Library	Platform
Escherichia coli RR1	SRR2014554	8192MB	24248885	Paired	Illumina HiSeq 2000
Escherichia coli 042	ERR007646	2406MB	14110696	Paired	(Illumina Genome Analyzer
Escherichia coli P12b	SRR2000272	1350MB	2990758	Paired	Illumina MiSeq
Escherichia coli KLY	SRR1424625	1682MB	6886668	Paired	Illumina HiSeq 2000
Escherichia coli O25b:H4-ST131	SRR933487	1070MB	3214312	Paired	Illumina Genome Analyzer IIx
Kineococcus rhizosphaerae DSM 19711	SRR6479489	2048MB	5641334	Paired	Illumina HiSeq 2500
Kineococcus xinjiangensis DSM 22857	SRR6479482	2168MB	5971022	Paired	Illumina HiSeq 2500
Mycobacterium tuberculosis F11	SRR974839	1936 MB	7279254	Paired	Illumina HiSeq 2000
Mycobacterium tuberculosis XDR KZN 4207	SRR1144800	1884MB	7033428	Paired	(Illumina HiSeq 2000
Arcobacter halophilus	SRR7587111	588MB	670813	Paired	454 Titanium
Rhodopirellula baltica	SRR7819959	1984MB	3207713	Single	Ion Torrent
Escherichia coli O157:H7 in Romania	ERR2375157	1201MB	2106268	Single	Ion Torrent
Rathayibacter tritici	SRR6799098	157MB	160403	Single	454 Junior
Salmonella enterica	SRR7905974	2990MB	163468	Single	PacBio
Staphylococcus aureus	SRR7739756	1336MB	86389	Single	Oxford nanopore MinIon
Pseudomonas aeruginosa	ERR2162181	246MB	1146696	Single	SoliD 5500

Table 5. Organisms and SRA number used to validate NGSReadsTreatment.

Organism	Coverage	Dataset Name
Mycobacterium bovis BCG str. Korea 1168P	300x	HS25MicoKorea1168P_300
Mycobacterium bovis BCG str. Korea 1168P	200x	HS25MicoKorea1168P_200
Mycobacterium bovis BCG str. Korea 1168P	100x	HS25MicoKorea1168P_100
Mycobacterium tuberculosis KZN 4207	300x	HS25MicoKZN_4207_300
Mycobacterium tuberculosis KZN 4207	200x	HS25MicoKZN_4207_200
Mycobacterium tuberculosis KZN 4207	100x	HS25MicoKZN_4207_100
Escherichia coli O103:H2 str. 12009	300x	HS25EcoliO103_H2_300
Escherichia coli O103:H2 str. 12009	200x	HS25EcoliO103_H2_200
Escherichia coli O103:H2 str. 12009	100x	HS25EcoliO103_H2_100

Table 6. Generation of simulated data with different coverage.

and the average memory usage by the program. The output was formatted using the -f option or the TIME environment variable. The string type format was interpreted in the same way as *printf*, where common characters were copied directly whereas special characters were copied using \t (tab) and \n (new line). The percent sign is represented by %% (otherwise, % indicates a conversion[13]).

Raw data download. Fastq-dump version 2.9.2 (https://edwards.sdsu.edu/research/fastq-dump/) was used to download the NCBI-SRA database files in fastq format.

Tool validation with real datasets. To validate NGSReadsTreatment were used data from sixteen organisms. Two strains of *Mycobacterium tuberculosis*, two *Kineococcus*, six strains of *Escherichia* coli, and one strain of *Rhodopirellula báltica*, *Arcobacter halophilus*, *Rathayibacter tritici*, *Salmonella entérica*, *Staphylococcus aureus* and *Pseudomonas aeruginosa*. Each organism with its SRA number is listed in Table 5. For paired reads the File size by Dataset and Total of Reads by Dataset represent the sum of tag1 and tag2 (Table 5).

Tool validation with simulated datasets. Aiming to further validate the tool NGSReadsTreatment another approach was employed, the use of simulated NGS datasets. The idea is that the tool NGSReadsTreatment should exhibit the same behavior in both real and simulated data.

To generate the simulated datasets, the ART tool version 2.5.8[9] was used, which is able to generate simulated next-generation reads from different platforms, based on a reference in the fasta format. The ART tool can simulate real sequencing read errors and quality, and it is used to test or benchmark a variety of method or tools for next-generation sequencing data analysis.

For this validation of the NGSReadsTreatment were simulated reads from sequencing on the Illumina HiSeq 2500 and Roche 454 GS FLX Titanium platforms.

The organisms used as reference to generate the simulated reads were: *Mycobacterium bovis* BCG str. Korea 1168P (GenBank: CP003900.2), *Mycobacterium tuberculosis* KZN 4207 (GenBank: CP001662.1), *Arcobacter halophilus* strain CCUG 53805 (GenBank: CP031218) and *Escherichia coli* O103:H2 str. 12009 (GenBank: AP010958.1). For each of the organisms two sets of reads were generated, one of the Illumina platform and another of the 454 platform.

Tool validation with simulated datasets of different coverage. A third validation step was performed, this time using simulated data with different sequencing coverage. The goal was to simulate different amounts of redundant reads by mimicking the PCR process. We selected as reference the genomes *Mycobacterium bovis* BCG str. Korea 1168P (dataset prefix name HS25MicoKorea1168P) *Mycobacterium tuberculosis* KZN 4207 (dataset prefix name HS25MicoKZN_4207) *and Escherichia coli* O103:H2 str. 12009 (dataset prefix name HS25EcoliO103_H2).

Each of the reference genomes was used in ART tool version 2.5.8 to generate simulated datasets with 100x, 200x and 300x coverage, respectively. Thus, nine simulated datasets were generated as shown in Table 6.

After this step we use an *ad-hoc* script (available in https://sourceforge.net/projects/ngsreadstreatment/files/AnalyzeDuplicatesInFastq.pl) designed to count the number of unique reads in a dataset, this is, reads that appear only once. The purpose of using this script was to determine if after processing the data, redundant reads were completely removed, thus ensuring that only unique reads would stay in each dataset. In this way, after each of the nine datasets were processed by each of the tools, the number of unique reads of each one was counted.

Workstation. The Workstation used to carry out the analyzes has the following configuration: Intel Core i7-2620M CPU 2.70 GHz with four processing cores, 324 GB HD and 6GB memory.

References

1. Reuter, J., Spacek, D. & Snyder, M. High-Throughput Sequencing Technologies. *Molecular Cell* **58**, 586–597 (2015).
2. Ebbert, M. *et al*. Evaluating the necessity of PCR duplicate removal from next-generation sequencing data and a comparison of approaches. *BMC Bioinformatics* **17** (2016).
3. Manconi, A. *et al*. Removing duplicate reads using graphics processing units. *BMC Bioinformatics* **17** (2016).
4. Xu, H. *et al*. FastUniq: A Fast De Novo Duplicates Removal Tool for Paired Short Reads. *PLoS ONE* **7**, e52249 (2012).
5. Burriesci, M., Lehnert, E. & Pringle, J. Fulcrum: condensing redundant reads from high-throughput sequencing studies. *Bioinformatics* **28**, 1324–1327 (2012).
6. Li, W. & Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22**, 1658–1659 (2006).
7. González-Domínguez, J. & Schmidt, B. ParDRe: faster parallel duplicated reads removal tool for sequencing studies: Table 1. *Bioinformatics* **32**, 1562–1564 (2016).
8. Expósito, R., Veiga, J., González-Domínguez, J. & Touriño, J. MarDRe: efficient MapReduce-based removal of duplicate DNA reads in the cloud. *Bioinformatics* **33**, 2762–2764 (2017).
9. Huang, W., Li, L., Myers, J. & Marth, G. ART: a next-generation sequencing read simulator. *Bioinformatics* **28**, 593–594 (2011).
10. Fan, B., Andersen, D., Kaminsky, M. & Mitzenmacher, M. Cuckoo Filter. *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies - CoNEXT '*14, https://doi.org/10.1145/2674005.2674994 (2014).
11. Bloom, B. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* **13**, 422–426 (1970).
12. Pagh, R. & Rodler, F. Cuckoo hashing. *Journal of Algorithms* **51**, 122–144 (2004).
13. Kerrisk, M. *The Linux programming interface*. (No Starch Press, 2010).

Acknowledgements

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). AAOV was supported by PRO2154-2018 from Federal University of Pará (UFPA) and PHCGS was supported by 092017-767 from Federal Rural University of Amazonia (UFRA).

Author Contributions

Antonio Sérgio Cruz Gaia developed the computational tool and article writing; Pablo Henrique Caracciolo Gomes de Sá reviewed the graphical interface and the article; Mônica Silva de Oliveira reviewed the tools and approaches, the user's manual; Adonney Allan O. Veras designed the project and review of the article.

Additional Information

Supplementary information accompanies this paper at https://doi.org/10.1038/s41598-019-48242-w.

Competing Interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.