

DIB (Drosophila Inversion Breakpoints) Pipeline

João Tomás Mota Cunha

Masters in Bioinformatics and Computational Biology
Computer Science Department
2023

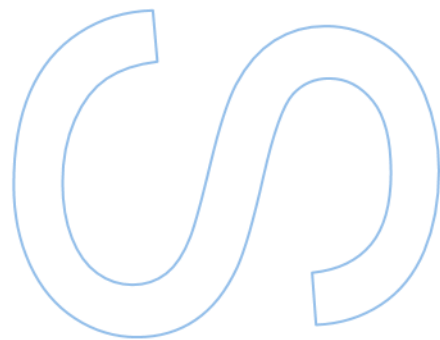
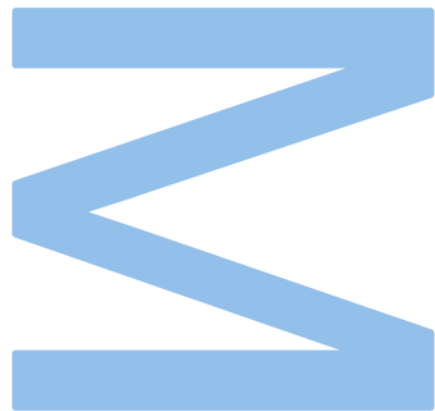
Supervisor

Jorge Manuel De Sousa Basto Vieira, Principal Investigator, IBMC,
i3S, UP
jbvieira@ibmc.up.pt

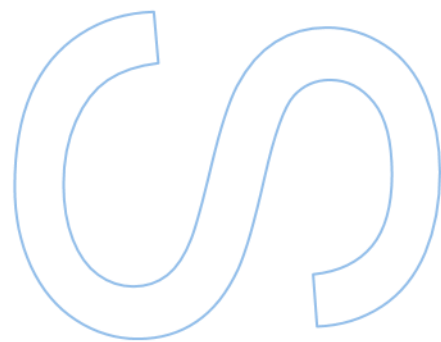
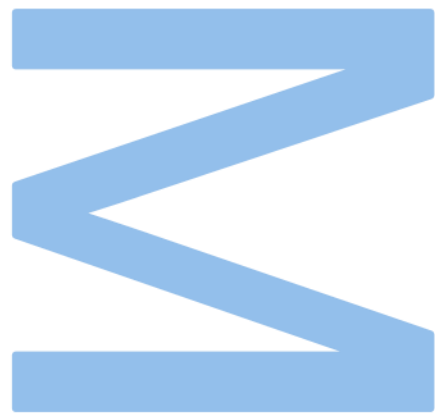
Co-supervisors

Cristina Alexandra Gonçalves Paula Vieira, Associate researcher,
IBMC, i3S, UP
cgvieira@ibmc.up.pt

Hugo López Fernández, Collaborator, i3S, UVigo
hugo.fernandez@i3s.up.pt



U. PORTO
FC FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO



Acknowledgements

I would like to express my sincere gratitude to the Masters in Bioinformatics and Computational Biology program, which provided me with all the necessary tools to overcome this great challenge that was the realization of my dissertation. And, I am especially thankful to my dedicated supervisors, Jorge Vieira, Cristina Vieira, and Hugo López-Fernández, for their unwavering support, guidance, and commitment throughout my Master's thesis journey. Their mentorship has been instrumental in shaping my academic growth.

I am, also, profoundly thankful to the entire team of the Phenotypic Evolution group for providing me with the invaluable opportunity to intern and collaborate, which significantly enriched my academic experience.

In addition, I owe a debt of gratitude to my family and friends for their steadfast support not only during this past year but throughout the entirety of my academic journey. I acknowledge that without your support and encouragement, this chapter of my life would not have reached its successful conclusion. I am deeply grateful for the pivotal roles each of you played in my academic journey.

Resumo

Esta tese, intitulada de “*DIB (Drosophila Inversion Breakpoints) Pipeline*”, apresenta uma abordagem abrangente e inovadora para a identificação e análise de pontos de inversão nos genomas de *Drosophila*, um grupo de organismos modelo amplamente utilizado na investigação genética. Ao longo desta investigação, foi desenvolvido um fluxo de trabalho completo que aborda todas as etapas críticas da análise de pontos de inversão, desde a aquisição de dados até à simulação, validação e análises avançadas de bioinformática. A pesquisa começa por detalhar os pré-requisitos essenciais, incluindo a obtenção de dados brutos do Sequence Read Archive (SRA) e o estabelecimento do ambiente local com o Docker. Além disso, é apresentado um script adicional que permite a criação de dados de sequenciação sintéticos com sequências genéticas personalizadas, aumentando a versatilidade do fluxo de trabalho.

No centro deste trabalho está a simulação de sequências a partir de um genoma de referência, um passo fundamental para gerar dados sintéticos que imitam de perto as verdadeiras sequências de sequenciação. Esta simulação permite comparações significativas entre dados sintéticos e experimentais, facilitando a identificação de potenciais pontos de inversão. Além disso, a personalização dos parâmetros de simulação proporciona aos utilizadores controlo sobre vários aspetos do processo.

Um dos destaques desta pesquisa é a ênfase na validação de pontos de inversão utilizando o Blastn, uma ferramenta fundamental de bioinformática. A aplicação de critérios de filtragem específicos para identificar os pontos de inversão mais relevantes é demonstrada, sendo fornecidos exemplos de como estes resultados podem ser utilizados para análises posteriores.

É crucial notar que, quando são utilizados dados reais de sequenciação, a interpretação dos resultados pode ser desafiadora devido à complexidade dos dados genómicos e à presença de variações genéticas. No entanto, o fluxo de trabalho fornece uma base sólida para enfrentar esses desafios.

Em resumo, esta pesquisa oferece um quadro robusto e uma ferramenta valiosa para a identificação e análise de pontos de inversão em *Drosophila*, contribuindo significativamente para a compreensão da genética deste organismo modelo. O fluxo de trabalho, juntamente com o script adicional, capacita os investigadores a explorar

cenários genéticos específicos e a aprimorar as suas análises, impulsionando a investigação neste campo crucial da genética.

Palavras-chave: *Drosophila*, Inversão, Ponto de inversão, Pipeline, genomas, bioinformática, fluxo de trabalho, dados, Docker, SRA, Blastn, simulação, validação, filtragem, script, análises.

Abstract

This thesis, titled “*DIB (Drosophila Inversion Breakpoints) Pipeline*” presents a comprehensive and innovative approach to the identification and analysis of inversion breakpoints in *Drosophila* genomes, a widely used model organism group in genetic research. Throughout this research, a complete workflow has been developed that addresses all critical stages of inversion breakpoint analysis, spanning data acquisition, simulation, validation, and advanced bioinformatics analyses. The research begins by detailing the essential prerequisites, including the procurement of raw data from the Sequence Read Archive (SRA) and the establishment of the local environment with Docker. Additionally, an additional script is introduced, allowing the creation of synthetic sequencing data with customized genetic sequences, enhancing the versatility of the workflow.

At the core of the work is the simulation of sequence reads from a reference genome, a fundamental step for generating synthetic data that closely mimics real sequencing reads. This simulation enables meaningful comparisons between synthetic and experimental data, facilitating the identification of potential inversion breakpoints. Furthermore, customization of simulation parameters provides users with control over various aspects of the process.

A key highlight of this research is the emphasis on the validation of inversion breakpoints using Blastn, a fundamental bioinformatics tool. The application of specific filtering criteria to identify the most relevant inversion breakpoints is demonstrated, and examples are provided of how these results can be used for further analyses.

It is crucial to note that when real sequencing data are employed, the interpretation of results can be challenging due to the complexity of genomic data and the presence of genetic variations. However, the workflow provides a solid foundation for addressing these challenges.

In summary, a robust framework and a valuable tool for the identification and analysis of inversion breakpoints in *Drosophila* are provided by this research, significantly contributing to the understanding of the genetics of this model organism. The workflow, combined with the additional script, empowers researchers to explore specific genetic scenarios and enhance their analyses, driving research in this crucial field of genetics.

Keywords: *Drosophila*, Inversion, Breakpoint, Pipeline, genomes, bioinformatics, workflow, data, Docker, SRA, Blastn, simulation, validation, filtering, script, analyses.

Table of Contents

List of Figures	ix
List of Abbreviations	x
1. Introduction	1
1.1. Chromosomal Inversions	1
1.2. Transposable Elements (TEs)	5
1.3. <i>DAIBAM</i> MITE Element	5
1.4. The identification of inversion breakpoints (TakeaBreak App)	6
1.5. Other valuable tools (WGSIM app)	9
1.6. Pipelines	9
1.7. Docker Images	10
1.7. Programming Language (Bash)	11
2. Aim of the work	13
3. Methodology	14
3.1. Databases	14
3.1.1. European Nucleotide Archive (ENA)	14
3.1.2. National Center for Biotechnology Information (NCBI).....	14
3.2. Software and Programs	15
3.2.1. AfterQC.....	15
3.2.2. Basic Local Alignment Search Tool (BLAST)	16
3.2.3. Wgsim.....	17
3.2.4. TakeaBreak.....	17
3.3. Automating Workflows with <i>DIB</i> Pipeline	18
3.3.1. Data Collection.....	18
3.3.2. Data Processing.....	18
4. Results and Discussion	20
4.1. <i>Drosophila</i> Inversion Breakpoints pipeline	20
4.1.1. <i>Drosophila</i> Inversion Breakpoints pipeline.....	20

4.1.2.	Databases Utilized	22
4.1.3.	Software and Tools	22
4.1.4.	Default Settings.....	23
4.2.	Getting Started	23
4.2.1.	Installation.....	23
4.2.2.	Prerequisites	23
4.2.3.	Configuring the Configuration File	24
4.2.4.	Running the Pipeline	25
4.2.5.	Reviewing Results.....	26
4.3.	Data Collection	26
4.3.1.	Acquiring SRA Data	26
4.3.2.	Acquiring Reference Genome	27
4.4.	Data Processing	27
4.4.1.	Overview	28
4.4.2.	Reference Genome Simulation	28
4.4.3.	Quality Control (QC).....	28
4.4.4.	File Format Conversion	29
4.4.5.	Inversion Breakpoint Detection.....	29
4.4.6.	Validation Through Blastn	29
4.5.	Simulating Reads	29
4.5.1.	Customizing Simulation Parameters.....	30
4.5.2.	Simulation Output.....	31
4.6.	Enhancing Functionality with an Additional Script	31
4.6.1.	Empowering User Creativity.....	32
4.6.2.	Flexible Parameterization	32
4.6.3.	File execution.....	32
4.6.4.	Usage example	33
4.7.	Identifying Inversion Breakpoints	34
4.7.1.	Overview	35

4.7.2.	Running TakeaBreak.....	35
4.7.3.	TakeaBreak Output	35
4.7.4.	Validation and Further Analysis.....	35
4.8.	Validation with Blastn.....	36
4.8.1.	Overview	36
4.8.2.	Filtering Blast Results.....	36
4.8.3.	Further Analysis	37
4.9.	Obtaining Results Files	37
4.9.1.	Output Format and Information	37
4.9.2.	Output Format and Information	38
4.9.3.	Usage with Real Sequences	39
4.10.	Troubleshooting.....	40
4.10.1.	Common Issues and Solutions	40
4.10.2.	Solutions.....	41
5.	Conclusions and Further Perspectives.....	42
	References	44

List of Figures

Figure 1) Schematic illustration describing how recombination is suppressed in an inversion heterozygote..... 1

Figure 2) Schematic illustration of chromosome inversion caused by ectopic recombination 3

Figure 3) Graphic illustration of the non-homologous end joining repair of the staggered breaks as the cause of chromosomal inversions..... 3

Figure 4) A visual representation of an inversion..... 7

Figure 5) Schematic illustration of an inversion pattern produced by the sequences alb (the blue path) and alb (the red path) in a Bruijn Graph with $k=4$ 8

Figure 6) *Drosophila* Inversion Breakpoints (*DIB*) Workflow 21

Figure 7) Example of a Configuration file 25

Figure 8) Config file Wgsim Options..... 30

Figure 9) *Drosophila melanogaster* Genome Assembly example 33

Figure 10) Standard values of the Wgsim employed in the “create_sim_file” Script.... 34

Figure 11) Example of a simulated file with Breakpoints at positions 400 and 450 within the Reference Genome..... 34

Figure 12) Config file Filtering Options 36

Figure 13) Example of a Fake Data Results file 37

Figure 14) Example of the folder with simulated input files 38

Figure 15) Example of the folder with simulated output files..... 39

Figure 16) Example of Real Data Results file 39

List of Abbreviations

FCUP	Faculty of Sciences of the University of Porto
UP	University of Porto
UVigo	University of Vigo
i3S	Institute of Investigation and Innovation in Health
<i>DIB</i>	<i>Drosophila</i> Inversion Breakpoints
DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
TIRs	Terminal Inverted Repeats
TPase	Transposase
TSDs	Target Site Duplications
NGS	Next Generation Sequencing
SAM	Sequence Alignment Map
BAM	Binary Alignment Map
SNPs	Single Nucleotide Polymorphisms
INDEL	Insertion and Deletion
Bash	Bourne Again shell
ENA	European Nucleotide Archive
NCBI	National Center for Biotechnology Information
DDBJ	DNA DataBank of Japan
SRA	Sequence Read Archive
EMBL	European Molecular Biology Laboratory
ANI	Average Nucleotide Identity
QC	Quality Control
Blast	Basic Local Alignment Search Tool
Blastn	BLAST for nucleotides
SRR	Sequence Read Archive Run Accession Number
RefSeq	Reference Sequence

1. Introduction

1.1. Chromosomal Inversions

Chromosomal inversions stand as rearrangements wherein a segment of a chromosome undergoes a reversal, end-to-end. These inversions are of two different types: pericentric inversions, which encompass the centromere, and paracentric inversions, which do not contain the centromere. A crossing-over event with one breakpoint within the inverted segment and another outside, in heterozygous individuals, produces imbalanced gametes carrying deletions, insertions, and in the case of pericentric inversions a count of either two or zero centromeres [1]. The impact of inversions on transmission dynamics was first highlighted by Sturtevant [2], a pioneer in genetic mapping, who observed the elimination of recombinant outcomes in heterozygous configurations, potentially reducing the fitness of heterozygotes (Figure 1).

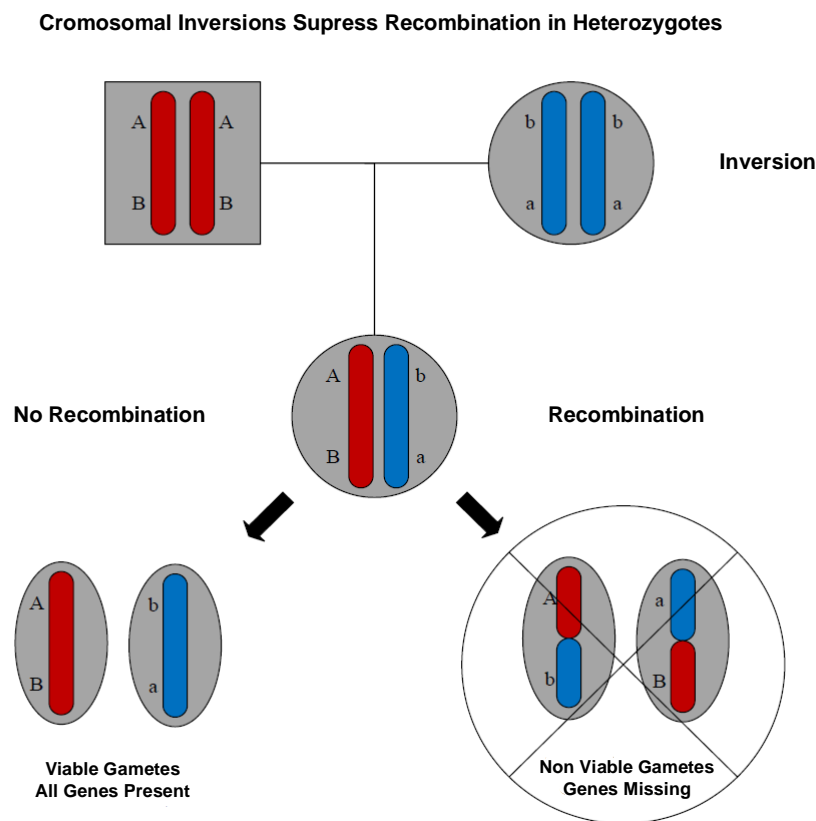


Figure 1) Schematic illustration describing how recombination is suppressed in an inversion heterozygote. Two loci segregate for the alleles (A, a) and (B, b), and an individual that is heterozygous at both loci and the inversions does not end up in A/b and a/A.

Within evolutionary dynamics, inversions are influenced by two principal factors: selection and random drift. Drift alone can lead to an increase in frequency of many inversions, particularly small intergenic ones [1]. Negative selection can impact the evolution of inversions in two ways: can impact inversions by causing meiotic functional anomalies (akin to certain pericentric inversions) or one of its breakpoints can disturb an open reading frame, changing gene expression [3]. This category encompasses a few human genetic disorders, see for instance [4]. Concurrently, positive selection can also effects the inversion itself, when it houses one or more favoured alleles [1].

Inversions play a significant role in the *Drosophila* genus due to reduced recombination in males and the tendency of aberrant recombinant outcomes to form polar body nurse cells in females [5]. This reduced fertility is less noticeable in this species, making it an ideal model for studying inversion formation of inversions, their increased frequency, and their influence on the genome. Because of the abundance of species in this genus, as well as the high likelihood of chromosomal inversions persisting, these organisms are choosier excellent models for studying these phenomena.

Within the family *Drosophilidae*, rearrangement rates vary across lineages and Muller's elements [6, 7]. However, the mechanisms governing inversion origination, their increased frequencies, and their genome-altering dominion remain enigmatic, a void that perpetuates the scarcity of comparative investigations encompassing species distantly related to *Drosophila melanogaster*.

Chromosomal inversions can emerge through chromosomal breakage followed by an erroneous repair of the homologous free-ends by nonhomologous end-joining (NHEJ) [8] (Figure 2). This is a common mechanism in *D. melanogaster* and closely related species [9]. Ectopic recombination (or nonallelic homologous recombination) between multiple copies of DNA sequences oriented in opposite directions within the same chromosome can also formed inversions (Figure 3).

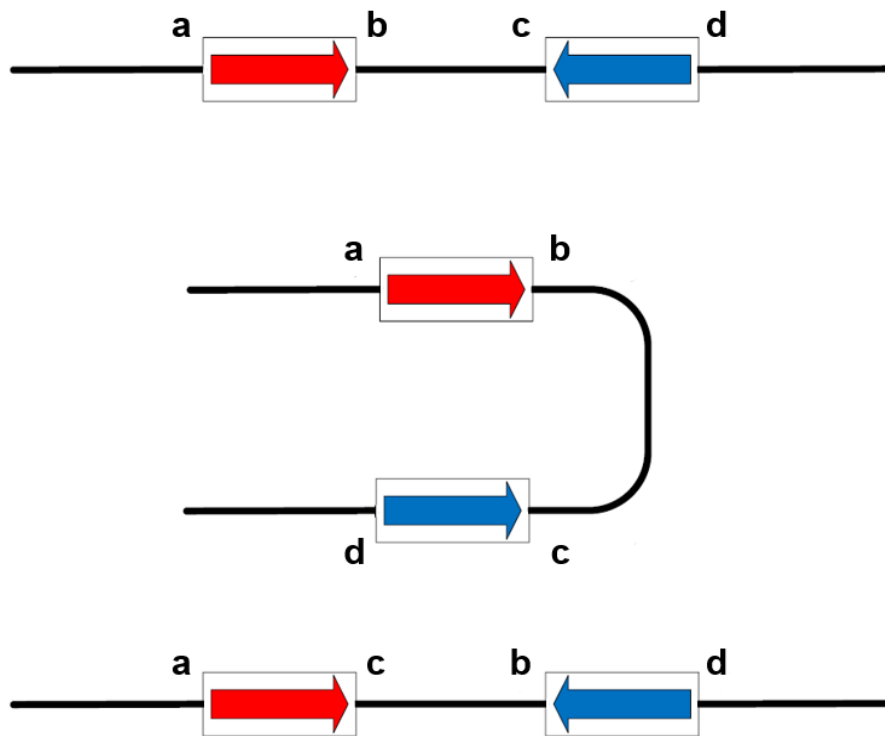


Figure 2) Schematic illustration of chromosome inversion caused by ectopic recombination.

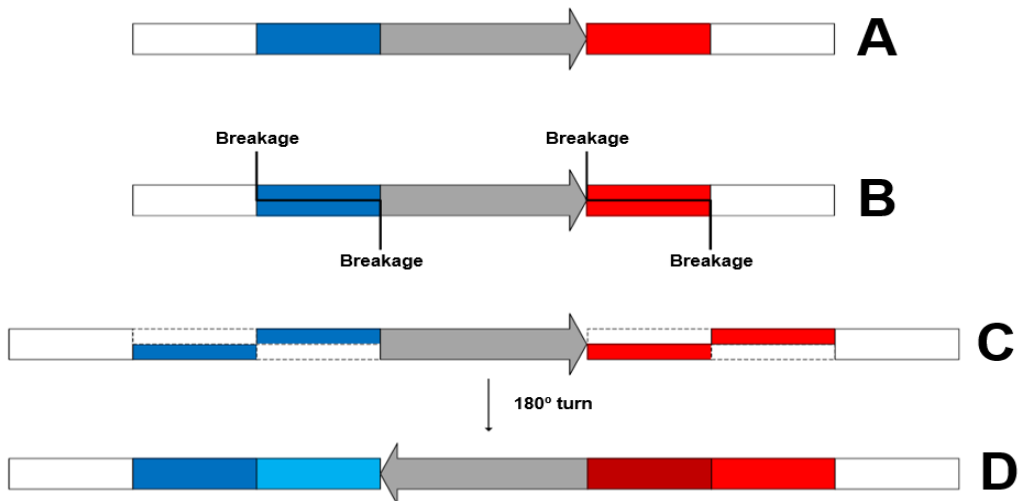


Figure 3) Graphic illustration of the non-homologous end joining repair of the staggered breaks as the cause of chromosomal inversions. **A)** A specific chromosomal region is flanked by two sequences that typically span single copy genes; **B)** The flanking sequences suffer stagger breaks that could lead to isochromatids or sister chromatids; **C)** Staggered breaks result in incorrect nitrogen base pairing, separating the sequences; **D)** The NHEJ mechanism forms the junction of the 5' end with the 3' end of the other breakpoint (in both breaks), producing 180° rotation of the relevant chromosomal region and duplication of the neighbouring sequences. The staggered breaks flanking the chromosomal areas are repaired. The inverted region and its direction are represented in this diagram by the gray arrow. In the top figure, adjacent sequences, which were initially single copies, are shown by dark blue on the left and light red on the right.

Apart from observations in *D. melanogaster* and closely related species [9], in the realm of *Drosophila*, the formation of polymorphic and fixed chromosomal inversions has been studied in other *Drosophila* species by examining their breakpoint sequences. Transposable elements (TEs), segmental duplications, and short repeat sequences play significant roles in shaping chromosomal inversions across various *Drosophila* species [10-13]. This interplay with TEs has been implicated in shaping chromosomal inversions across various *Drosophila* species including: *D. buzzatii* [10, 14-16], *D. americana* [5, 11, 12, 17], *D. virilis* [11], *D. novamexicana*, *D. lummei*, *D. mojavenensis*, and *D. uniseta*. DAIBAM (*Drosophila americana* inversion breakpoints associated MITE) emerges as a dominant catalyst, steering at least 89% of the observed chromosomal inversions within and amidst *virilis* phylad species [9].

The rate of fixation of chromosomal inversions is known to vary across the *Drosophila* phylogeny, but it is unclear whether this variation is due to differences in the rate at which chromosomal inversions are generated or to varying selection pressures. Studies on the origin of polymorphic chromosomal inversions and of fixed inversions between closely related species are likely to be the most informative regarding this issue. Indeed, when looking at the breakpoints of very old inversions it may be difficult to recognize the remains of a TE since it is neutrally evolving, and thus the TE may be lost in time. It should be said that in *Drosophila* the rate of deletion is high compared with other species [18]. It was discovered that the relative rates of insertions compared to point substitutions are relatively similar, as are the average sizes of insertions. In *Drosophila*, however, deletions are both more common and much larger than in *mammals*. *Drosophila* had a 2.6 times higher rate of deletions per point substitution than *mammals*, and the average deletion sizes were roughly eight times greater in *Drosophila*. *Drosophila* eliminates DNA approximately 20-fold faster than *mammals* due to its higher rate of creation and larger average size of deletions, losing nonessential DNA at a pace that is approximately 60 times that in *mammals*. As a result, lineages with high rates of DNA loss should have small, "tidy" genomes with few pseudogenes and short introns, whereas lineages with low rates of DNA loss should have big, "messy" genomes with high amounts of "junk" DNA of various types [18]. The high rate of DNA loss is prevalent and probably ancestral for all *drosophilids* [19].

It is likely that in the future more *Drosophila* genomes will become available for closely related species. When this happens it will be finally possible to get a good estimate of the fraction of inversions that are originated by ectopic recombination between

transposable elements, and what is the prevalent mode of chromosomal inversion generation [5].

1.2. Transposable Elements (TEs)

Transposable Elements (TEs) are mobile DNA sequences present in both eukaryotes and prokaryotes, capable of moving within the genomic landscape. These elements come in two types: retrotransposons, that mobilize through an intermediary RNA molecule, and transposons, that are mobilized via a single or double-stranded DNA counterpart [20]. Transposons are further classified into three categories based on their transposition mechanisms: cut-and-paste, rolling circle, and Mavericks [21]. Cut-and-paste transposons, characterized by terminal inverted repeats (TIRs), typically have short TIRs and encode transposase (TPase), a protein responsible for their excision from the original genomic location. Subsequently, they are then reinserted into new sites, resulting in the creation of target site duplications (TSDs) [22]. Among the distinguished cut-and-paste transposon families, P [23] and mariner [24] hold prominence in *Drosophila*, standing as prominent examples within a broader array of transposon super-families.

The introduction of harmful mutations and adaptation can be facilitated by mobile genetic elements [25]. TEs are known to influence genome evolution in a variety of ways, including modifying expression patterns and gene structure, as well as genome size and organization, which contributes to chromosome rearrangement [26, 27].

At the population level, TEs are most often found as individual copies, with their presence limited to low frequencies, often below 5%. This is the result of a balance between transposition rate and the selective pressures that mount against isolated TE copies, especially as TEs become more prevalent [28].

In the context of chromosomal inversions, particularly within the *virilis* phylad, nonallelic homologous recombination between TEs emerges as the predominant mechanism underlying their formation [5, 11].

1.3. DAIBAM MITE Element

The DAIBAM (*Drosophila americana* inversion breakpoints associated MITE) element plays a predominant role in the formation of chromosomal inversions. It emerges as a Miniature Inverted-Repeat Transposable Element (MITE) and has been implicated in

eight out of the nine reported chromosomal inversions spanning *virilis* phylad species (*D. virilis*, *D. lummei*, *D. novamexiana*, and *D. americana*) [5, 11].

These findings strongly suggest that TE-mediated ectopic recombination, particularly involving *DAIBAM*, is the dominant mechanism driving the development of inversions in this species group. It's noteworthy that *DAIBAM* has only been identified in species within the *virilis* group so far. However, it is plausible that this TE is a rapidly evolving MITE and hence difficult to identify in distantly related species [11].

A study conducted by Delprat et al. [10] provided substantial evidence demonstrating the role of ectopic recombination triggered by *Galileo* TEs in the formation of three polymorphic *D. buzzatii* inversions.

1.4. The identification of inversion breakpoints (TakeaBreak App)

Structural variations in genomes are vital sources of genetic diversity, contributing to phenotypic differences, inherited diseases, and evolutionary processes. The extent of structural variations in populations has been only recently acknowledged, thanks mainly to Next Generation Sequencing (NGS) [29].

The identification of inversion breakpoints directly from sequencing reads or even assembled genomes is not a trivial task. Therefore, two bioinformatics approaches have been used to identify them, that are here reviewed. In *Drosophila*, Fonseca et al. [11] and Reis et al. [5] used a genome alignment approach using MAUVE [30] followed by a confirmation using *blastn*. This seems to be an efficient approach that produce few false positives but requires assembled genomes [31]. Or the main approach discussed by Medvedev et. al. that calls structural variant breakpoints when mapped read pairs show discordant mappings with respect to expected insert-size and orientation of the reads, in the reference genome [32]. Due mainly to repetitions in complex genomes and mapping errors, these methods suffer from high false positive rates and a small overlap between predictions obtained by different methods [31].

However, these methodologies are fundamentally constrained by their reliance on reference genomes and one or both genomes being comparted. This issue gets worse when we employ species without a high-quality reference genome.

This raises the question whether it is even possible to find inversion signatures directly in raw NGS reads without the need for a reference genome or full read assembly. This

question can be answered using Lemaitre et al. analysis [33] and formal modeling of topological patterns formed by de Bruijn Graph inversions. Their analysis led to the development of the TakeaBreak app [33] which incorporates an algorithm that detects these inversion patterns, decoding inversion breakpoints directly from NGS reads. It avoids the crutch of reference genomes and the process of genome “reassembly” in contrast to its predecessors. This application circumvents these constraints, offering a solution that is not only reference-free but also computationally efficient. It utilizes extremely little memory and analyses data quickly, allowing it to run on standard machines and achieve an acceptable execution time. For instance, when simulating Illumina reads with a 2x40x coverage of the human chromosome 22, this tool efficiently processes the data within a span of 2 hours, harnessing less than 1GB of memory (<https://github.com/GATB/TakeABreak/blob/master/README.md>).

A specific motif in the de Bruijn Graph, such as that in Figure 1, can be generated to produce an inversion pattern. Only the $k-1$ k-mers spanning each inversion breakpoint separate the two sequences, accounting for any other variations in terms of k-mers between the two sequences, with and without the inversion.

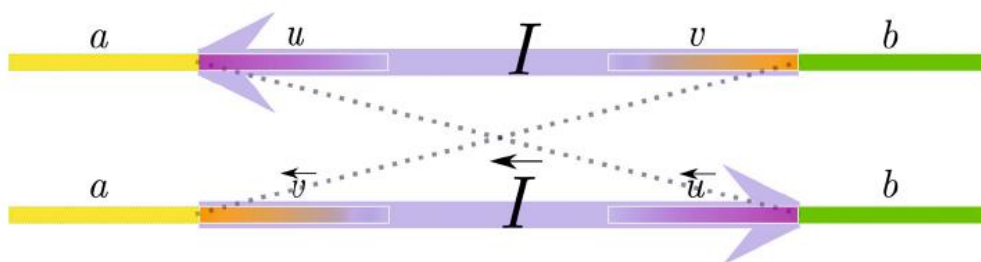


Figure 4) A visual representation of an inversion. The intersections of the inverted segment and the non-inverted segments, which result in two breakpoints for each sequence, are the four inversion breakpoints. The sequences alb and $a\bar{l}b$, display the breakpoints for the four specific k-mers a , u , v , and b . Adapted from [33].

The de Bruijn Graph's fork, which is defined by a common node $n\bar{a}$ that branches to two separate k-paths that terminate in $n\bar{u}$ and $n\bar{v}$, respectively, characterizes the breakpoints at the left of the inverted segments. The other two breakpoints (at the right) were similar in that they were identified by two k-paths that originate from $n\bar{u}$ and $n\bar{v}$ that join in $n\bar{b}$. The de Bruijn Graph is formed by these two forks, which are linked by two common nodes that represent the k-mers u and v , respectively, and their reverse complements. This pattern, which is referred to as the inversion pattern, is shown in Figure 5.

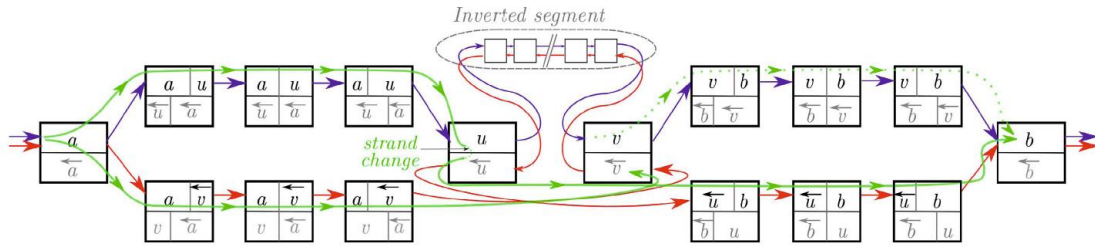


Figure 5) Schematic illustration of an inversion pattern produced by the sequences alb (the blue path) and alb (the red path) in a Bruijn Graph which $k=4$. Nodes are shown as two-stage boxes with the explicit kmer shown in the upper portion, which is colored black, and the implicit k-mer shown in the lower part, which is colored grey. Instead of representing DNA k-mers, the node content displays the fraction of the four main k-mers a, u, v, b and their reverse complements. In order to keep things simple and maintain generality, we assume that all k-mers of au, vb, $a\bar{v}$ and $u\bar{b}$ are explicitly stored. The state of a node traversed by edges entering and leaving its upper (resp. lower) part is explicit (resp. implicit). The paths described by the TakeABreak algorithm are represented by the green paths. Only when n_v and n_b have been found is the dashed green path checked. Adapted from [33].

This motif has four k-paths in the de Bruijn graph that can travel through it: one from $n\bar{a}$ to $n\bar{u}$; one from $n\bar{u}$ to $n\bar{b}$, one from $n\bar{b}$ to $n\bar{v}$; one from $n\bar{v}$ to $n\bar{a}$. A de Bruijn graph only has fixed length pathways, making it simple and rapid to identify the presence of such a pattern.

It's important to note that this pattern has several flaws. In the beginning, it looks for inversion breakpoints but does not actually perform the inversion. Another drawback is the motif's completely symmetrical, which results in the same inversion being found whether you start at node $n\bar{b}$, or $n\bar{u}$ or $n\bar{v}$.

Because it does not produce more k-mers or routes in the de Bruijn Graph than the non-inverted sequence, an inversion will not arise if it has an inverted repetition of size greater than or equal to $k - 1$ at its breakpoints. For instance, if $a = \bar{b}$ or $u = \bar{v}$, this is true. The inversion pattern is still produced even if an inverted repetition has a length less than $k - 1$, although it is not entirely symmetrical.

Lemaitre's et al. [33] experiments and simulations were made to demonstrate the reliability of the TakeaBreak pattern and algorithm, which can yet be improved to better fit actual genome re-sequencing data. In the conducted experiments, TakeaBreak demonstrated its high efficiency to identify inversion breakpoints in this case by predicting the 1000 true positive inversions, yielding a 100% recall for 100% precision. However, it's important to note that the test dataset was relatively small, based on the bacterial *Escherichia coli* K12 genome. In this dataset, 1000 random inversions were simulated, correctly found. In fact, the breakpoints were not affected by any new polymorphism, only inversions were simulated. Rearrangement distributions are likely, not random, and some rearrangements may be connected to repeated sequences, as opposed to inversions, which were given a uniform distribution. The simulations only

included entire blunt-ended breakpoints, which may not accurately reflect the molecular mechanisms underlying such events. For instance, non-homologous end joining, a common DNA repair mechanism, can result in tiny insertions or deletions near the breakpoint, which were not simulated in this study.

1.5. Other valuable tools (WGSIM app)

The bioinformatics toolkit boasts SAMtools [33] a versatile library and software package highly regarded in genomics analysis. Its capabilities extend to the analysis and editing of Sequence Alignment Map (SAM) and Binary Alignment Map (BAM) files. It was designed to develop the complex landscape of genomics [34]. BAM files contain alignment information of sequences that are mapped to reference sequences. BAM files are the compressed binary version of SAM files, making them smaller and more efficient to work with when compared to SAM files.

The tool known as WGSIM available in SAMtools, can be used to simulate reads from diploid genomes rich in Single Nucleotide Polymorphisms (SNPs) and insertion and deletion (INDEL) polymorphisms. It creates a canvas of sequences covered with variations, with each SNP and INDEL. This tool also allows the creation of sequencing errors, adding realism to the simulated reads [35]. Moreover, it records the true read coordinates, number of polymorphisms, and sequencing errors in the read names. One of WGSIM's notable advantages arises when a FASTA file needs to be incorporated into a mapping pipeline that exclusively accepts FASTQ files (for example, variant calling with closed genomes).

1.6. Pipelines

The development of efficient pipelines is critical in the field of bioinformatics, automating analyses and time-consuming processes involving the use of various types of tools, data or even applications. In this context, a pipeline is a series of interrelated processing steps that transform raw data into relevant insights. Bioinformatics researchers work with massive amounts of genomics, transcriptomics, proteomics, and other techniques datasets. Analysing this data necessitates a complex set of computing operations, including data pre-treatment, quality control, alignment, variant calling, and statistical analysis, among others [36].

Manually carrying out each step can be time-consuming and error-prone, particularly when dealing with huge datasets. This is where pipelines come in: they allow researchers

to define and automate these steps, decreasing human intervention and the chance of errors. Researchers obtain various benefits from utilizing pipelines. To begin with, automation eliminates the need for repetitive manual interventions, allowing scientists to concentrate on higher-level research and interpretation. This speeds up the overall research process, allowing enormous datasets to be processed in a fraction of the time it would take manually. Furthermore, pipelines promote consistency in data processing, eliminating variability caused by human mistake.

Additionally, pipelines promote research repeatability and transparency. The pipeline defines and documents each step of the analysis, from data pre-treatment to the final outcome. This ensures that peers and collaborators may more easily comprehend and repeat the analysis, increasing the research's credibility.

Bioinformatics analyses can be computationally demanding, taking hours or even days to complete. Pipelines efficiently distribute jobs by exploiting parallel processing and cluster computing capabilities to accelerate calculations. This reduction in processing time allows researchers to swiftly iterate through analyses, refine ideas, and investigate new paths of enquiry.

1.7. Docker Images

Understanding Docker images is essential when working in Docker to build applications in containerized environments (<https://docs.docker.com/get-started/overview/>). A Docker image unfolds as a potent entity, a digital tool that unlocks the realm of code execution within a Docker container. This image is made up of a set of files that group together all the essential elements (such as installations, application code, environment variables, dependencies, and libraries) needed to set up a fully operational container environment [37].

The Dockerfile method is particularly favoured in the field of bioinformatics. This approach promises organization, flexibility, and efficiency in the endeavour to construct Docker images. This, in turn, becomes the key to create compact, reliable and secure container environments [38].

The rewards of containerized application deployment manifest aplenty:

- Performance:** Docker images are extremely user-friendly because they simply require the installation of Docker. This ease of use allows researchers and developers to get up

and running quickly, without having to deal with complicated setup procedures or compatibility difficulties.

•**Portability:** Docker images are designed to be portable across several computing environments. This mobility is revolutionary for anyone who needs to collaborate or seamlessly shift their work between desktops or cloud instances. The image contains not just the software but also its dependencies, ensuring consistent functionality regardless of where it is executed.

•**Access to Repositories:** Docker images can be securely saved in public repositories such as Docker Hub [39]. This accessibility facilitates the sharing and distribution of scientific software among researchers.

•**Cross-Platform Compatibility:** Docker images are versatile. They can be run on both Linux, Windows and Mac platforms. This cross-platform compatibility eliminates the hassles caused by software compatibility concerns between different operating systems. It ensures that your scientific program operates consistently across a wide range of computing environments.

•**Pipeline Integration:** Docker images can connect effortlessly into pipelines. Docker containers provide a consistent and reproducible environment for each step of your pipeline, whether you're processing massive datasets or coordinating complex operations. This facilitates collaboration and sharing of challenging data processing and analysis tasks.

1.7. Programming Language (Bash)

Bash, short for “Bourne Again shell” is a scripting language that translates sets of instructions into seamless examples of automation and control. Essentially, Bash serves as a command interpreter, bridging the gap between human intent and machine execution. This scripting language works in tandem with the command line interface, allowing users to encapsulate command sequences into cohesive scripts that may be performed with a single command [40].

With Bash, scripts can be written that accomplish a variety of activities. From the simple to the complex, its syntax and constructs offer the foundation for navigating directories, manipulating files, performing computations, and even deal with complex operations [41]. Bash scripts go from simple lines of text to strong tools capable of manipulating the command line by exploiting variables, conditions, loops, and functions.

Furthermore, Bash can interactively prompt users for input, dynamically allocate memory, and gracefully handle failures through its error handling features. Its capacity to redirect input and output, concatenate instructions through pipelines, and manipulate strings and arrays significantly expands the realm of possibilities from the command line. A well-designed Bash script can automate mundane operations, expedite intricate procedures, and simplify data processing, liberating human efforts for more creative pursuits [40].

In essence, “bash” stands as a versatile and robust tool, empowering users to construct intricate codes within the confines of the command line. Bash has evolved from its humble origins as a command interpreter to become a cornerstone of automation, allowing users to harness the potential of the command line interface and execute their digital activities with precision and finesse [42].

2. Aim of the work

The primary objective of this research is to establish a seamless and comprehensive pipeline capable of automating the identification of inversion breakpoints. This innovative approach relies on the integration of TakeaBreak and pool-seq data, setting itself apart by its self-sufficiency, operating without the need for the scaffolding of a reference genome and bypassing the process of genome “reassembly”.

However, it's crucial to acknowledge the inherent limitations of this tool. Its efficacy may be compromised, especially in scenarios involving the formation of inversions facilitated by transposable elements, a phenomenon notably observed in the *virilis* group.

This endeavour represents one among a series of pipelines under development. Each approach within this series possesses unique strengths and weaknesses, offering researchers a diverse array of user-friendly tools for the discovery of inversion breakpoints.

3. Methodology

3.1. Databases

In this study, two crucial databases were employed:

3.1.1. European Nucleotide Archive (ENA)

The European Nucleotide Archive (ENA) serves as a public repository for nucleotide sequence data and plays a pivotal role as a founding member and partner in the International Nucleotide Sequence Data Collaboration. This longstanding data exchange initiative collaborates with the National Center for Biotechnology (NCBI) [43] and the DNA DataBank of Japan (DDBJ) [44].

This public archive provides an open and supported platform for data management, archiving, publication, and data dissemination [45]. ENA comprises three main databases: the Sequence Read Archive (SRA), the Trace Archive, and the EMBL Nucleotide Sequence Database (EMBL-Bank). In this study, we utilized the SRA database as an archival resource for next-generation sequences [46].

The SRA database functions as a repository of raw sequence data with the aim of balancing the long-term archival cost with the necessity to store sufficient information to support the reuse of submitted data. Data submitted to SRA is organized using a metadata model consisting of six objects: study, sample, experiment, run, analysis, and submission. The SRA experiment and run objects, used in this study, contain library and instrument information and are directly linked to the sequence data. The SRA analysis object is employed for the deposition of various analysis results, including alignments and assemblies [47].

3.1.2. National Center for Biotechnology Information (NCBI)

The National Center for Biotechnology Information (NCBI), situated within the National Library of Medicine at the National Institutes of Health, was established to develop information systems for molecular biology [43].

NCBI, which manages the GenBank database [48], collects and curates prokaryotic type strains and their genomes, referred to as 'type assemblies.' These type assemblies serve as unambiguous references for taxonomic names and play a crucial role in comparative genomics, particularly when calculating average nucleotide identity (ANI) and verifying and reclassifying a taxon [49].

The NCBI Assembly Resource (<https://www.ncbi.nlm.nih.gov/assembly/>) enables users to search the Assembly database to locate genome assemblies of interest. The Assembly database stores the names and identifiers for the sequences in each genome assembly and records the organization of the component sequences into scaffolds and chromosomes. This facilitates reporting the assembly structure and providing mappings between names, synonyms, and identifiers for assemblies, chromosomes, or scaffolds. The Assembly database also maintains various metadata about genome assemblies, including names, dates, assembly completeness, the group responsible for generating the assembly, and details about the sequenced organism and sample [50].

As with individual sequence records, when referencing an assembly, it is crucial to include the version; otherwise, the sequence content of the assembly will be undefined. For instance, assemblies in GenBank have accessions prefixed with GCA, for instance, “GCA 000002285.2”, while those in RefSeq have accessions prefixed with GCF, for instance, “GCF 000002285.2”, as used in this study.

The RefSeq project's provision of curated and stable annotated reference genomes, transcripts, and proteins for selected viruses, microbes, organelles, and eukaryotic organisms has enabled researchers to focus on the best representative sequence data, in contrast to redundant data in GenBank, and to unambiguously reference specific genetic sequences. The RefSeq collection offers explicitly linked genome, transcript, and protein sequence records that incorporate publications, informative nomenclature, and standardized and expanded feature annotations [51].

3.2. Software and Programs

For the execution of this work, it was also necessary the intervention of some applications, such as there listed below.

3.2.1. AfterQC

AfterQC [52] is a next-generation sequencing (NGS) tool used in the field of genomic data refinement. This application operates on FASTQ raw data and encompasses a range of functionalities, including automatic filtering, trimming, error correction, and data quality control. AfterQC is designed to process FASTQ files in batch mode, streamlining the analysis of data from sequencing runs involving multiple samples.

When processing FASTQ files, AfterQC follows a rigorous quality control (QC) and filtering pipeline. For each input, whether it's a single FastQ file or a pair of files, AfterQC generates high-quality reads. [52]

This tool was specifically developed to address common challenges associated with sequencing data quality control and filtering. In addition to standard quality control functions, such as per-cycle base content and quality statistics, AfterQC introduces innovative features like automatic trimming and overlapping analysis.

For the use of this application only the default values were used in all the options offered (<https://github.com/OpenGene/AfterQC>). These default values were deemed sufficient to fulfil the study's objectives, rendering further adjustments unnecessary.

3.2.2. Basic Local Alignment Search Tool (BLAST)

The Basic Local Alignment Search Tool (BLAST) is a widely used bioinformatics algorithm and software tool, allowing researchers to compare a query sequence, whether it's a protein (amino acid) or nucleotide (DNA/RNA) sequence, with a vast database of sequences to identify similar or homologous sequences within that database [53]. BLAST's key functionality includes:

- **Sequence Comparison:** BLAST allows the comparison of a target sequence (the query) against a reference database to find regions of similarity or identity.
- **Threshold-Based Matching:** BLAST uses a scoring system to identify sequences in the database that share significant similarity with the query sequence above a specified threshold. This threshold can be adjusted to control the stringency of the search.
- **Diverse Applications:** BLAST has a wide range of applications in molecular biology and genomics. It can be used for tasks such as identifying the species of an unknown sequence, locating specific functional domains within a protein, mapping DNA sequences to a reference genome, and annotating genes and their functions.

BLAST provides various programs tailored for specific sequence types and comparison scenarios, the one used in this work is specifically **Nucleotide-nucleotide BLAST (blastn)**; is a specific program within the BLAST suite that is designed for comparing DNA (nucleotide) query sequences against a DNA sequence database.

While numerous parameters and customization options are available in BLAST (<https://www.ncbi.nlm.nih.gov/books/NBK279684/>), the standard values were kept unchanged for this work since they did not require modification for the specific research purposes.

3.2.3. Wgsim

Wgsim [54], available as part of the SAMtools package [55], is a versatile program that excels in simulating diploid genomes enriched with Single Nucleotide Polymorphisms (SNPs) and insertion and deletion (INDEL) polymorphisms. This tool constructs a dynamic landscape of sequences adorned with genetic variations, incorporating both SNPs and INDELS. Moreover, Wgsim empowers users to introduce sequencing errors, thereby enhancing the authenticity of the simulated reads [34].

Wgsim beyond simulating polymorphisms also records essential details such as the true read coordinates, the count of polymorphisms, and any sequencing errors within the read names. Providing, particularly when integrating a FASTA file, a mapping pipeline that exclusively accepts FASTQ files, as seen in applications like variant calling for closed genomes. All options/settings available when using this tool are available in (<https://manpages.debian.org/stretch/samtools/wgsim.1>).

It is worth noting that the majority of these settings adhere to standard configurations. For the purposes of this study, there was no necessity to deviate from these default values.

3.2.4. TakeaBreak

TakeaBreak is a tool that detects inversion breakpoints directly from NGS (Next Generation Sequencing) reads, without the use of any reference genome and without the “reassembly” of the genomes. This application uses little memory and analyses data quickly, allowing it to run on standard machines and achieve an acceptable execution time. This approach is what distinguishes it from the others, consisting of the use of fixed-size topological patterns within the Bruijn graph [33].

TakeaBreak offers a range of adjustable parameters (as you can see in <https://github.com/GATB/TakeABreak/blob/master/README.md>), covering the creation of De Bruijn graphs, the breakpoint detection algorithm, and various settings related to computational runtime and memory usage. For this work, default parameter values were

retained to ensure the tool's accessibility and usability, avoiding an excessive burden of specific parameter tuning for end-users while still achieving reliable results.

3.3. Automating Workflows with *DIB* Pipeline

3.3.1. Data Collection

The workflow starts with the download of Sequence Read Archive (SRA), FASTQ files, specifically those containing “short reads” generated by high-throughput sequencing, typically less than 1,000 base pairs in length. These files are obtained through automatic downloads from the European Nucleotide Archive (ENA) database, facilitated by a provided list of **Run Accession Numbers** from the user.

Simultaneously, the workflow harnesses the NCBI Assembly database, which houses information about the structural makeup of assembled genomes. A genome assembly, in essence, is a computational representation of a genome sequence. Given the limitations of sequencing entire chromosomes in one go, each chromosome assembly comprises short, sequenced DNA segments expertly stitched together. During this pipeline step, the user needs only supply the NCBI **RefSeq Assembly number** to access the reference FASTA file.

In both data collection steps, users have the option to incorporate personal files, encompassing both SRR and reference files. This flexibility empowers users to work with unpublished files, extend their search within the database, or even experiment with synthetic data.

3.3.2. Data Processing

The initial tool deployed in this pipeline is AfterQC, employed on each SRR file to conduct quality control checks on the resultant FASTQ files. Following this verification, every FASTQ file is transformed into a FASTA file format, optimizing memory and space utilization throughout the computational process.

For the downloaded reference file, a distinct procedure unfolds, invoking the Wgsim tool to simulate FASTQ sequence reads from a reference genome FASTA file. In this step, users have the opportunity to simulate diploid genomes featuring SNPs, INDELS, as well as readings with uniform substitution sequencing errors.

Subsequently, the TakeaBreak application is utilized to identify potential inversion breakpoints directly from Next Generation Sequencing (NGS) reads. This is achieved exclusively using the SRR input files (post-quality control) and the simulated files.

This phase yields a FASTA file containing the putative inversion breakpoints, necessitating a validation step involving Blastn.

To perform this validation, it was necessary to perform some necessary conditions in order to filter only the information relevant to the final goal. Delimiting through minimum values of “**Sequence Identity**”, “**Sequence Size**”; such as only accepting inversions that have the “**4 Sequence Types**”, and a certain number of “**Genomic Hits**”.

Following these operations, a “Results File” is obtained, housing each potential inversion breakpoint, accompanied by additional contextual information, including precise position details.

4. Results and Discussion

While TakeaBreak demonstrates proficiency in identifying potential inversion breakpoints, it falls short in providing researchers with readily accessible and user-friendly insights. Nevertheless, this innovative pipeline takes a significant stride forward by seamlessly integrating TakeaBreak with additional cutting-edge tools and methodologies.

Through the optimization of potential inversion breakpoints, this pipeline streamlines the workflow for scientists, providing a more transparent pathway to recognizing and confirming these breakpoints within the reference genome. Moreover, it empowers researchers to extract valuable evolutionary insights from the genetic data.

In the following sections, we delve into the mechanics of this pipeline, explore its advanced methodologies, and discuss the consequential findings.

4.1. *Drosophila* Inversion Breakpoints pipeline

User Manual for the *Drosophila* Inversion Breakpoints Workflow: This user guide offers comprehensive instructions for harnessing a specialized bioinformatics pipeline aimed to identify potential inversion breakpoints in *Drosophila* genomes, using FASTQ files. One can observe a graphical representation of this pipeline in Figure 6.

4.1.1. *Drosophila* Inversion Breakpoints pipeline

The primary goal of this workflow is to automate the detection of inversion breakpoints in *Drosophila* genomes using next-generation sequencing (NGS) data. Inversions represent critical genetic variations with profound implications for an organism's characteristics and evolutionary trajectory. This pipeline streamlines NGS data analysis, generates simulated reads, and identifies potential inversion breakpoints with precision.

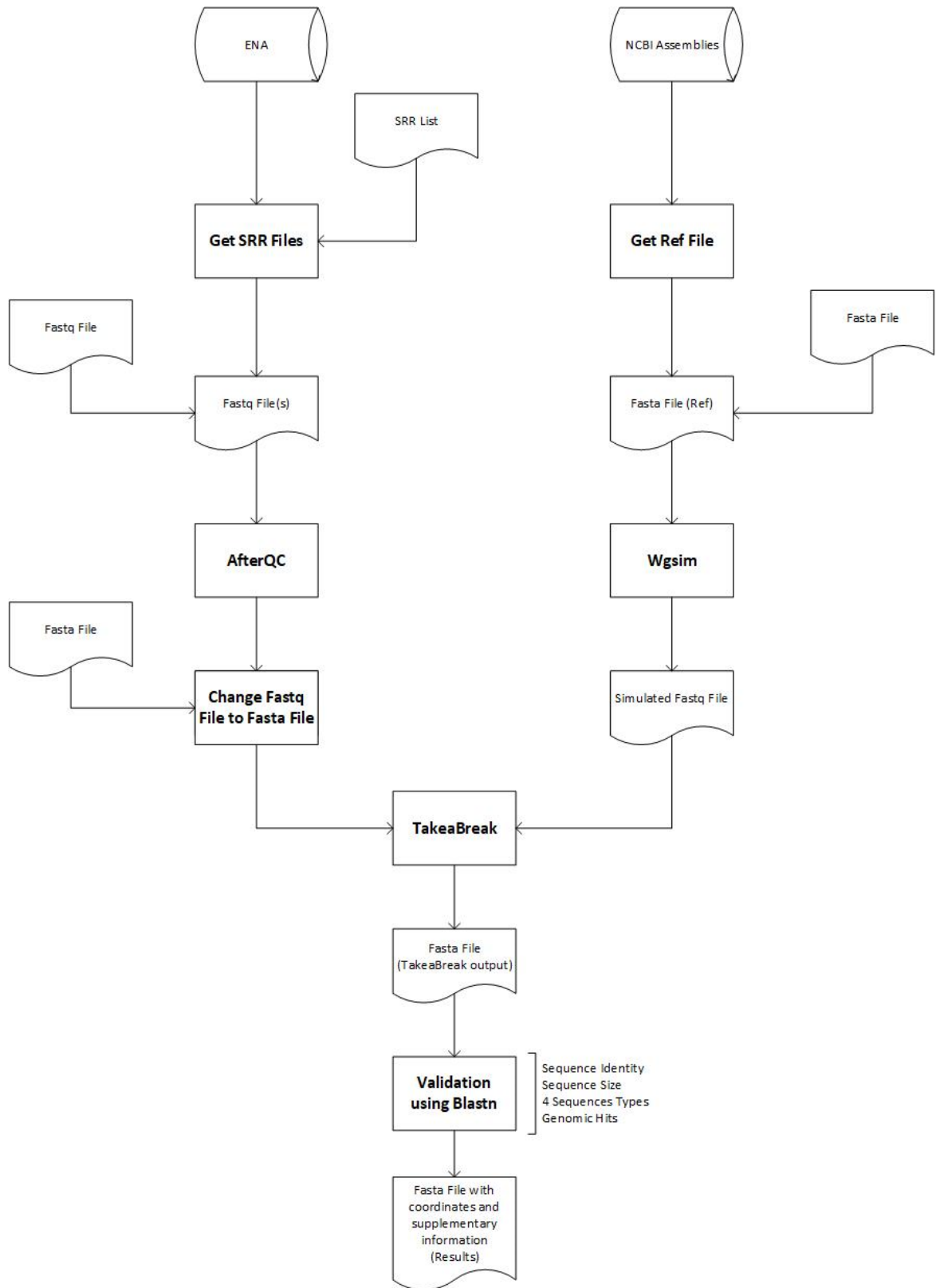


Figure 6) *Drosophila* Inversion Breakpoints (DIB) Workflow.

4.1.2. Databases Utilized

This pipeline relies on two essential databases to accomplish our research goals:

- **European Nucleotide Archive (ENA):** ENA is a well-established public repository for nucleotide sequence data. We specifically utilize the Sequence Read Archive (SRA) within ENA to access raw sequence data relevant to this study.
- **National Center for Biotechnology Information (NCBI):** NCBI stands as a stalwart institution in the realm of molecular biology research. Manages the GenBank database and, in this study, is an important tool to access genome assemblies and comprehensive details about the sequenced organisms and samples of interest.

4.1.3. Software and Tools

This workflow employs several specialized bioinformatics tools and programs, including:

- **Wgsim:** An integral component of the SAMtools package, Wgsim excels at generating simulated diploid genomes enriched with genetic variations. It enables the creation of dynamic sequence landscapes complete with single nucleotide polymorphisms (SNPs), insertions and deletions (INDELs), and sequencing errors.
- **AfterQC:** versatile tool tailored for refining next-generation sequencing (NGS) data. This tool automates filtering, trimming, error correction, and quality control of FASTQ data, ensuring the utilization of only high-quality reads in the analysis. The tool operates in batch mode, facilitating the processing of multiple samples simultaneously.
- **TakeaBreak:** A unique tool specifically designed for detecting inversion breakpoints directly from NGS reads, without relying on the assembly of a reference genome. Its efficiency and rapid data analysis capabilities make it a valuable asset within the pipeline.
- **BLAST (Basic Local Alignment Search Tool):** A widely adopted bioinformatics algorithm and software tool employed to compare nucleotide sequences. In this pipeline, BLAST's nucleotide-nucleotide (blastn) program is harnessed to identify sequences of interest in the analysis.

4.1.4. Default Settings

To ensure accessibility and user-friendliness across varying expertise levels, the default settings for the tools and programs integrated into this pipeline have been predominantly retained. These default values have been thoughtfully selected to align seamlessly with the goals of this research, minimizing the need for extensive parameter adjustments.

4.2. Getting Started

This section will provide users with a step-by-step guide to initiate the *Drosophila* Inversion Breakpoints Workflow. Prior to beginning, it is crucial to ensure access to the necessary data and resources outlined in the pipeline's methodology. Familiarity with the databases, software, and tools introduced in the preceding sections is essential.

4.2.1. Installation

Local Environment: Docker is a fundamental requirement for this workflow. It provides a consistent and isolated environment for running bioinformatics tools. You can install Docker on your system by following the comprehensive guidelines available in the Bioinformatics Docker Images Project (<https://pegi3s.github.io/dockerfiles/>) or consulting the Docker Manual (<https://docs.docker.com/get-started/overview/>). Docker is compatible with all major operating systems, including Linux, Windows, and Mac.

Pipeline Code: To access the *Drosophila* Inversion Breakpoints Workflow, you'll need to obtain the Docker image containing the pipeline code. Use the “**docker pull pegi3s/dib**” command to retrieve the image from the Bioinformatics Docker Images Project (<https://pegi3s.github.io/dockerfiles/>) [38]. Ensure that you have the latest version of the code to benefit from performance enhancements and access the most recent features.

4.2.2. Prerequisites

Before utilizing the pipeline, it is imperative to verify the presence of the following prerequisites:

SRR Data: Obtain the list of Run Accession Numbers (SRR) from the European Nucleotide Archive (ENA) database, which hosts the NGS data for analysis. Alternatively, users may choose to incorporate personally obtained or generated data in

either FASTA or FASTQ format. Remember to store this list in the project/main directory. If you possess pre-existing SRR files, ensure they are placed within a folder named “SRR_data” also located in the project/main directory.

Reference Genome: Identify the NCBI RefSeq Assembly number corresponding to the relevant reference genome. Alternatively, if users possess a pre-existing reference genome dataset stored in a folder named “reference_file” they can utilize their custom FASTA file. Make sure that the “reference_file” is also located in the project/main directory.

4.2.3. Configuring the Configuration File

Before initiating the pipeline, it's essential to configure the “config” file, as exemplified in Figure 7 (see below), to define preferences and provide essential details. The following parameters should be adjusted within the “config” file:

- **dir:** Set the directory path to the project directory.
- **get_reference:** Specify “Y” or “y” if automatic downloading of the reference genome is required; choose “N” or “n” if the reference genome is already available.
- **reference:** If “Y” or “y” is chosen for (**get_reference**) the NCBI RefSeq Assembly number should be provided for the automatic download of the reference genome. If “N” or “n” is selected for (**get_reference**) the filename of the reference genome should be provided. The reference genome should be stored in a folder named “reference_file” within the project directory.
- **list:** Indicate the filename containing the list of SRR numbers corresponding to the data. This list file must be located in the project directory.
- **format:** Choose either “fastq” or “fasta” based on the format of the SRR data.
- **Wgsim options:** Detailed information about these options can be found in the "Simulating reads" step.
- **Filtering Options:** Comprehensive details regarding filtering options are outlined in the “Validation with Blastn” step.

```
2 # Project Directory
3 dir=/home/evolution3/Documents/Joao_Cunha/pipeline
4
5 # Reference options
6 get_reference=n
7 reference=GCF_000001215.4_1200pb.fna
8
9 # SRR options
10 list=SRR_list.txt
11 format=fastq
12
13 # Wgsim options
14 sequence_size1=300
15 sequence_size2=300
16 rate_mutations=0.010
17 fraction_indels=0.15
18 prob_indel=0.30
19 error_rate=0
20 standard_deviation=50
21 read_pairs=1000000
22 seed=0
23
24 # Filtering options
25 seq_ident=97.000
26 align_length=50
27 gen_hits=5
```

Figure 7) Example of a Configuration file.

4.2.4. Running the Pipeline

To execute the pipeline, access the terminal and navigate to the project directory. The pipeline will sequentially execute the various steps outlined in the flowchart, including data collection, data processing, inversion breakpoint detection, and result filtering.

To run the pipeline you should adapt and run the following command: **docker run -v /var/run/docker.sock:/var/run/docker.sock -v /your/data/dir:/data pegi3s/dib bash -c ./main**

Make sure to replace “/your/data/dir” with the actual path to your project's main directory. This directory will receive the output files (results) generated by the pipeline.

4.2.5. Reviewing Results

Upon successful completion of the pipeline, the final results can be found in the project directory. These results will be presented in files named (e.g., “results.SRRnumber.fasta”) containing potential inversion breakpoints alongside additional contextual information.

4.3. Data Collection

The *Drosophila* Inversion Breakpoints Workflow initiates with the fundamental step of data collection. This phase involves the aggregation of requisite sequence data from publicly accessible repositories and, optionally, the inclusion of personal data files. The following section provides comprehensive instructions on data acquisition.

4.3.1. Acquiring SRA Data

To procure the SRA data necessary for your analysis, follow these steps:

- 1) Visit the European Nucleotide Archive (ENA):** Open your web browser and navigate to the European Nucleotide Archive (ENA) website at (<https://www.ebi.ac.uk/ena>).
- 2) Search for Sequencing Runs (SRR):** Utilize the ENA's search functionality to identify the sequencing runs (SRR) linked to your research project. Compile a list of SRR numbers corresponding to the datasets you plan to analyze.
- 3) Create an SRR List:** Create a text file (e.g., “SRR_list.txt”) and record each SRR number on a separate line. Save this list within your project directory. Alternatively, if you possess pre-existing files downloaded or generated, place them inside a folder named “**SRR_data**” within your project directory.
- 4) Configure the Configuration File:** Locate and open the “config” file situated within your project directory. Adjust the ‘list’ variable to match the filename of your SRR list (e.g., list=“SRR_list.txt”). In the case of pre-existing files within the “**SRR_data**” folder, no modifications are needed; the pipeline will automatically process all files within the folder.

4.3.2. Acquiring Reference Genome

The acquisition of the reference genome constitutes a pivotal step in the workflow. If you already possess the reference genome file, you may skip this section. Otherwise, follow these instructions:

- 1) Visit the National Center for Biotechnology Information (NCBI):** Launch your web browser and navigate to the National Center for Biotechnology Information (NCBI) website at (<https://www.ncbi.nlm.nih.gov/>). Select the **Assembly Database**.
- 2) Search for NCBI Assembly Number (RefSeq):** Use the NCBI's search functionality to find the reference genome affiliated with your research project. Once you've identified, copy the **NCBI RefSeq Assembly Number** (e.g., GCF_000001215.4).
- 3) Configure the Configuration File:** Open the config.sh file in your project directory. Set the "get_reference" variable to "Y" or "y" to signify your intention to automatically download the reference genome. Additionally, set the "reference" variable to match the **NCBI RefSeq Assembly Number** of your reference genome (e.g., "reference= GCF_000001215.4"). If you have already downloaded or created the reference file in the "**reference_file**" folder, set the "get_reference" variable to "N" or "n" and specify the file name of your reference genome (e.g., "reference= GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna").

By successfully acquiring the SRA data and the reference genome, you have completed the data collection phase of the *Drosophila* Inversion Breakpoints Workflow. These datasets will serve as the foundation for subsequent data processing and analytical stages.

4.4. Data Processing

After acquiring the necessary sequence data, the next phase in the *Drosophila* Inversion Breakpoints Workflow involves data processing. This stage employs various tools and procedures to ensure the quality and suitability of the raw data for subsequent analysis. The following section outlines the key steps involved in data processing.

4.4.1. Overview

Data processing encompasses the following essential tasks:

- **Reference Genome Simulation:** Simulating FASTQ sequence reads from the reference genome to generate synthetic data for comparative purposes.
- **Quality Control (QC):** Ensuring that the sequence data maintains high quality by identifying and rectifying issues such as low-quality reads and sequencing errors.
- **File Format Conversion:** Transforming the raw FASTQ files sourced from the Sequence Read Archive (SRA) into FASTA format to optimize memory and storage efficiency.
- **Inversion Breakpoint Detection:** Using the TakeaBreak application to identify potential inversion breakpoints from processed Next Generation Sequencing (NGS) reads.
- **Validation Through Blastn:** Confirming the inversion breakpoints by querying nucleotide sequences within a nucleotide database based on the provided reference genome, using Blastn.

4.4.2. Reference Genome Simulation

Simulating FASTQ sequence reads from the reference genome is a crucial step. This process generates synthetic data for comparison with experimental outcomes.

Wgsim Tool: As part of the SAMtools package, Wgsim is employed for simulating diploid genomes enriched with genetic variations, including SNPs and INDELS. Users have the flexibility to introduce sequencing errors, thereby enhancing the authenticity of the simulated reads. Wgsim meticulously records crucial details, such as true read coordinates and the count of polymorphisms, within the read names.

4.4.3. Quality Control (QC)

Prior to the initiation of any analysis, ensuring the quality of your sequence data is paramount.

AfterQC Tool: This tool is instrumental in conducting quality control assessments on the FASTQ files. It automates filtering, trimming, error correction, and data quality control procedures. AfterQC operates proficiently in batch mode, rendering it suitable for

handling multiple samples concurrently. The tool generates high-quality reads and generates QC profiling and HTML reports for each input file.

4.4.4. File Format Conversion

To optimize memory and storage utilization, the raw FASTQ files undergo conversion into FASTA format. This conversion process is indispensable for efficient data management.

4.4.5. Inversion Breakpoint Detection

The TakeaBreak application is harnessed to detect potential inversion breakpoints directly from NGS reads. This stage involves the scrutiny of post-quality control SRR files in conjunction with the simulated data.

4.4.6. Validation Through Blastn

The validation of inversion breakpoints is pivotal within the *Drosophila* Inversion Breakpoints Workflow. Following the identification of potential inversion breakpoints using the TakeaBreak application, the subsequent phase entails their validation via Blastn. Blastn is a widely adopted bioinformatics tool, employed for the juxtaposition of nucleotide sequences against a reference database, with the purpose of identifying homologous sequences.

With these data processing steps completed, you are prepared to move on to the analysis phase of the workflow, where potential inversion breakpoints will be identified and validated.

4.5. Simulating Reads

Simulating sequence reads from a reference genome is a foundational step in the *Drosophila* Inversion Breakpoints Workflow. This critical procedure enables the generation of synthetic data that closely mimics actual sequencing reads, providing a valuable resource for subsequent analysis and validation. In this section, we will walk you through the process of simulating reads from the reference genome.

4.5.1. Customizing Simulation Parameters

You have the flexibility to modify the simulation by adjusting various settings, as shown in the Figure 8.

```
#Advanced options (WGSIM app)
sequence_size1=300
sequence_size2=300
rate_mutations=0.010
fraction_indels=0.15
prob_indel=0.30
error_rate=0
standard_deviation=50
read_pairs=1000000
seed=0
```

Figure 8) Config file Wgsim Options.

Here are some key parameters, with default values indicated in brackets:

- **sequence_size1:** Specifies the length of the first read. [70]
- **sequence_size2:** Specifies the length of the second read (for paired-end sequencing). [70]
- **rate_mutations:** Sets the mutation rate. [0.0010]
- **fraction_indels:** Defines the fraction of indels (insertions/deletions) in the simulated reads. [0.15]
- **prob_indel:** Sets the probability of an indel happening. [0.30]
- **error_rate:** Specifies the sequencing error rate. [0.020]
- **standard_deviation:** Sets the standard deviation of the fragment size. [-1]
- **read_pairs:** Determines the number of read pairs to generate. [1000000]
- **seed:** Provides a random seed for reproducibility. [-1]

Five noteworthy parameters warrant special attention:

- **read_pairs:** This parameter may necessitate adjustment based on the desired size of the simulated file. It affords users the flexibility to tailor the output to meet specific file size requirements.
- **seed:** Setting this value to 0 guarantees replicable results when using the same input. This ensures consistency in outcomes during the analysis.
- **error_rate:** An error rate value of 0 implies the absence of base errors, resulting in perfectly accurate simulated sequences. However, users can modify this rate to model sequencing errors, mirroring those observed in real-world data.

Increasing the Base Error Rate introduces a higher incidence of base errors in the simulated reads, rendering the data more reflective of actual sequencing data with inherent imperfections. It's worth noting that, in the context of Illumina's reversible terminator technology, a recommended error rate of 0.1% is often considered (primarily encompassing substitution errors, with insertions/deletions occurring very rarely) [56].

- **sequence_size1** and **sequence_size2**: These parameters determine the size or length of the simulated sequencing reads. In Illumina sequencing contexts, the recommended read length is typically 300 base pairs, which is why these values have been utilized. Setting the same length for both parameters allow the creation of sequences suitable for paired-end sequencing. (<https://www.illumina.com/science/technology/next-generation-sequencing/planning-experiments/read-length.html>)

Please note that the **-d** (Outer Distance Between the Two Ends) and the **-h** (Haplotype Mode) settings are omitted, as they are considered irrelevant for this particular type of work.

4.5.2. Simulation Output

After executing Wgsim with your chosen parameters, the tool will generate synthetic FASTQ files containing simulated reads. These files serve as invaluable resources for subsequent analysis and validation within the workflow. Armed with these simulated reads, you are prepared to advance to the subsequent stages of the workflow, including the detection of potential inversion breakpoints and their validation through Blastn.

4.6. Enhancing Functionality with an Additional Script

In our pursuit of maximizing the utility of our comprehensive pipeline, an auxiliary script was developed during the testing phase. This script serves as a valuable extension, seamlessly integrating with the Wgsim application, and endowing users with the ability to generate synthetic sequencing data featuring customizable genetic sequences adorned with inversions. The primary purpose of this script is to facilitate the creation of simulated genetic data, providing users with the freedom to design genetic sequences that incorporate breakpoints at specific positions of their choosing.

4.6.1. Empowering User Creativity

At its core, this supplementary script is engineered to streamline the creation of simulated genetic data, granting users the creative liberty to design genetic sequences infused with breakpoints at positions of their own choosing. Whether your research demands the exploration of specific genetic scenarios or necessitates the generation of tailored test data, this script is a versatile tool at your disposal.

4.6.2. Flexible Parameterization

To cater to diverse user needs, this script offers a spectrum of parameter options. Users can harness the full range of parameters provided by the Wgsim application, affording granular control over the simulation process. However, we've also simplified the user experience by implementing default values that suffice for most basic testing scenarios. With this simplified option, users need only specify their preferred reference file and define the positions for the two breakpoints they intend to introduce into the genetic sequence.

4.6.3. File execution

Upon execution, this script orchestrates the creation of an output file aptly named "inverted_simulated_reads". This file harbors the synthetic genetic data meticulously designed to include the user-defined breakpoints. Executing the script is a straightforward endeavor, as demonstrated in the following code:

```
docker run -v /var/run/docker.sock:/var/run/docker.sock -v /your/data/dir:/data  
pegi3s/dib bash -c “./simulate_data <reference_file> /your/data/dir  
<first.breakpoint.pos> <second.breakpoint.pos> <options>”
```

In this command, you should replace:

- **/your/data/dir** to point to the directory that contains the FASTA file you want to analyze. (You should adapt in the two places of the code)
- **<reference_file>** with the name of the input FASTA file you want to analyze (placed in the main/project directory).
- **<first.breakpoint.pos>** with the number of the position of the first breakpoint
- **<second.breakpoint.pos>** with the number of the position of the second breakpoint.

- **<options>** with the specific options of the Wgsim tool. You can substitute this parameter by just the letter “s”, meaning that you want to use the standard values represented in the Figure 10. Otherwise, you need to put the Wgsim parameters manually, the ones already explained above; In other words, you can add the options, by decrepit order: **sequence_size1**, **sequence_size2**, **rate_mutations**, **fraction_indels**, **prob_indel**, **error_rate**, **standard_deviation**, **read_pairs**, **seed**.

4.6.4. Usage example

In this illustrative example, we employed the “inverted_simulated_reads” as the foundation, using the reference genome “GCF_000001215.4_1200pb.fna”. In this instance, the reference genome encompasses the initial 1200 base pairs of the *Drosophila melanogaster* Genome Assembly, a characteristic illustrated in Figure 9.

```

1 >NC_004354.4 Drosophila melanogaster chromosome X 1200pb
2 GAATTCGTCAGAAATGAgctaacaatttaaatcattaaatgcGAGCGGCGAATCCGGA
3 AACAGCAACTTCAAACAGTCACTCTGGCTGAACTAAATGGCCTGATAAACTCACTGGAA
4 TTAAAGAAAGCCCCAGGAACTGACAATCTTAACAACAAGACCATAATAAACTTACCTACA
5 AAGGCCAgaatatatttaatacttATTTATAACAACATCCTGAGAACTGGACATTTCCCG
6 AACAAATGGAAGCACGCTAGCATCTCAATGATTCCCAAACAGGGAAATCACCATTTGCT
7 CTAAATTCATACCGCCAATCAGCTTACTCTCTGGTCTTTCCAAACACTCGAAAAGAATA
8 CTAAGAAACGACTGTATGACATTGACTCTTTTGCCAAAGCAATCCCTTCCCATCAATTT
9 GGTTTCAGAAAGGATCATGGAGCGGAACATCAGCTGGCCAGGGTGACCAATTTATTCTA
10 AAAGCTTTTGATGAAAAAGATGTCTGTTCTGCCACATTCCTTGACATTACGGAAGCCTTT
11 GACCGAGTATGGCAGCAGCGCTTGCTATATAAACTATCCAGACTCATCCCCAGATACCTA
12 TTCGACCTACTTGAAAACATTTTATCTAATAGAACCTTCTCAGTAAGGATCGACGGTGAA
13 ACAACGTCTAGGATAGGTAATATTAGAGCAGGAGTGCCCCAGGGCAGCATACTGGGACCG
14 GTCCTCTACTCAATATACTCATCCGACATGCCCTATCCCATCGTAAAAGACTATATGCGT
15 AACATATCCTTCCCTGATTACCACCCAATAATATTATCTTAGCTACATATGCAGATGAT
16 ACCATAATTCTTAGCCGGTCCAAATATACCAAGCTTGCGATCAACCTAAATCAAACACTAC
17 CTTAACGTCTTCTGTAGGTGGTCAAAAAAATGGGACATAGCaattaatgcaaaaaaaCC
18 GGACACATTCTTTTCTCcttaaaaaaagaacaaactAATATATACACTCCCCACTAATC
19 AACGGACAAAGAGCTGCCAAACTAAACAACAACGCTATCTCGGACTTATGCTAGACAGA
20 AGACTGACCTTTTGTGCACACATGACGCTGCTAAAGGGAAAGACTATAGCTGCATATAAA
21 AACTGGAATGGCTAATAGGAAAAACAGCCACCTacccaaaaatgcaaaaattctCCTC

```

Figure 9) *Drosophila melanogaster* Genome Assembly example.

The code used to run this simple example was:

```

docker run -v /var/run/docker.sock:/var/run/docker.sock -v
/home/evolution3/Documents/Joao_Cunha/pipeline/ficheiros_sim:/data
pegi3s/dib bash -c “./simulate_data GCF_000001215.4_1200pb.fna
/home/evolution3/Documents/Joao_Cunha/pipeline/ficheiros_sim 400 450 s”

```


computing hardware. In this section, we will provide guidance on utilizing TakeaBreak to identify inversion breakpoints.

4.7.1. Overview

The identification of inversion breakpoints is exclusively executed using the SRR input files (post-quality control) and the simulated reads generated in previous steps. TakeaBreak conducts an analysis of NGS data, leveraging fixed-size topological patterns within the De Bruijn graph. This approach renders TakeaBreak a potent tool for this specific task.

4.7.2. Running TakeaBreak

Upon executing the command, TakeaBreak initiates the analysis of the input data to identify inversion breakpoints. The tool utilizes the SRR input files (post-quality control) and the simulated reads generated in previous steps. Notably, this analysis is executed independently of the presence the assembly of a reference genome, rendering it a valuable tool for situations involving non-model organisms or when a reference genome is unavailable.

4.7.3. TakeaBreak Output

Following the completion of the analysis, TakeaBreak generates an output file containing putative inversion breakpoints. Typically, this output materializes as a FASTA file that incorporates the sequences corresponding to the breakpoints, along with supplementary contextual information.

4.7.4. Validation and Further Analysis

The output furnished by TakeaBreak serves as a starting point for the identification of potential inversion breakpoints. The subsequent phase within the workflow entails the validation of these putative breakpoints through Blastn. This involves the application of filtering criteria to select the most pertinent candidates. Armed with the knowledge of validated inversion breakpoints, you can proceed to corroborate them and amass additional insights concerning their genomic locations.

4.8. Validation with Blastn

The validation of inversion breakpoints is a crucial phase within the *Drosophila* Inversion Breakpoints Workflow. Once potential inversion breakpoints have been identified using the TakeaBreak application, the next step involves validating these breakpoints through Blastn. Blastn, a widely used bioinformatics tool, is employed to compare nucleotide sequences with a reference database, allowing for the identification of homologous sequences.

4.8.1. Overview

The primary goal of this step is to validate the putative inversion breakpoints identified by TakeaBreak. Blastn plays a vital role in confirming the presence of these breakpoints within the reference genome, revealing previously undisclosed sequences, and establishing evolutionary relationships (homology) between the input sequences and known genes.

4.8.2. Filtering Blast Results

To pinpoint valid inversion breakpoints, specific filtering criteria can be applied to the Blast results. These criteria may include, in the configuration file, the parameters sequence identity (**seq_ident**), alignment length (**align_length**), and the maximum number of genomic hits (**gen_hits**). This filtering process helps identify the most relevant candidates for further analysis. For instance, consider Figure 12, which represents the validation with Blastn step. It highlights cases where sequence identity exceeds 97%, alignment length is greater than 50 base pairs, and the number of hits in the reference genome is less than 5. Only cases where four sequence types are identified are considered valid for analysis.

```
#Filtering options  
seq_ident=97.000  
align_length=50  
gen_hits=5
```

Figure 12) Config file Filtering Options.

4.8.3. Further Analysis

After the filtering process, you will have a set of validated inversion breakpoints, accompanied by additional information about their genomic positions. These validated breakpoints are candidates for more in-depth study and analysis in your research.

4.9. Obtaining Results Files

In this section, we will delve into how the pipeline filters and processes data to ultimately derive the results files. These files are a pivotal outcome of the *Drosophila* Inversion Breakpoints Workflow, serving as the basis for further downstream analyses. We will elucidate the methodology employed and provide examples of the output format, elucidating the specific information encapsulated within the final files.

4.9.1. Output Format and Information

In this section, we'll provide an example of a final FASTA file representing a detected inversion with two breakpoints. This output highlights two breakpoints indicating their locations in the query sequences and whether they were found in the reference genome. This example illustrates the format and content of a final FASTA file, showing the positions in the query, and providing specific parameters used for filtering the Blastn results. (Figure 13)

```
1 >INV_a-u_0_rep_0 Not found in reference genome
2 CGACTGTATGACATTGACTCTTTTGCCAAAATGTTCCGCTCCATGATCCTTTCTGAAACC
3 >INV_v-b_0_rep_0 Not found in reference genome
4 TCTGAAACCAAATTGATGGGAAGGGATTGCCAGCTGGCCAGGGTGACCCAATTTATTCTA
5 >INV_a-vbar_0_rep_0 NC_004354.4 100.000 60 370 429 Found in reference genome
6 CGACTGTATGACATTGACTCTTTTGCCAAAGCAATCCCTTCCCATCAATTTGGTTTCAGA
7 >INV_ubar-b_0_rep_0 NC_004354.4 100.000 60 421 480 Found in reference genome
8 GGTTCAGAAAAGGATCATGGAGCGGAACATCAGCTGGCCAGGGTGACCCAATTTATTCTA
```

Figure 13) Example of a Fake Data Results file.

In this simple example:

- Every inversion event corresponds to four distinct entries within the FASTA file: The first two entries pertain to the breakpoint sequences, denoted as “**INV_a-u_0**” and “**INV_v-b_0**” which are anticipated to be present in one genome. Subsequently, the final two entries represent the corresponding breakpoint sequences in the other genome, designated as “**INV_a-vbar_0**” and “**INV_ubar-b_0**”

b_0". Additionally, the size of the exact repeat detected at the breakpoints is indicated in each entry header (e.g., "**rep_0**").

- "**NC_004354.4**" is the reference genome where these breakpoints were found.
- "**100.000**" indicates a 100% match or identity between the query and reference sequences.
- "**60**" is the length of the alignment or sequence overlap.
- "**370**" and "**421**" are the start positions of the alignment in the query for the first breakpoint, aligning precisely in the middle of their designated points, securely nestled at the position 400 base pairs.
- "**429**" and "**480**" are the end positions of the alignment in the query for the second breakpoint, perpetuating the precision of centering at the position 450 base pairs within the query.

Note: It is essential to note that these results were obtained using simulated inversion data for testing purposes. As you transition to real datasets and more complex research scenarios, the final FASTA files will contain information about genuine inversion breakpoints within your experimental data, but the resulting file may be more complex.

4.9.2. Output Format and Information

Even when utilizing simulated data, there exists the possibility of deviations from expected results. Such deviations can be attributed to various factors, as elucidated in the forthcoming topic "Troubleshooting". The creation of this Fake Data Results file involved the simulation of sequences with breakpoints at various positions, mirroring the diversity observed in Figure 14. In Figure 15, the output obtained from this test was significantly condensed, taking into account the input files.



Figure 14) Example of the folder with simulated input files.

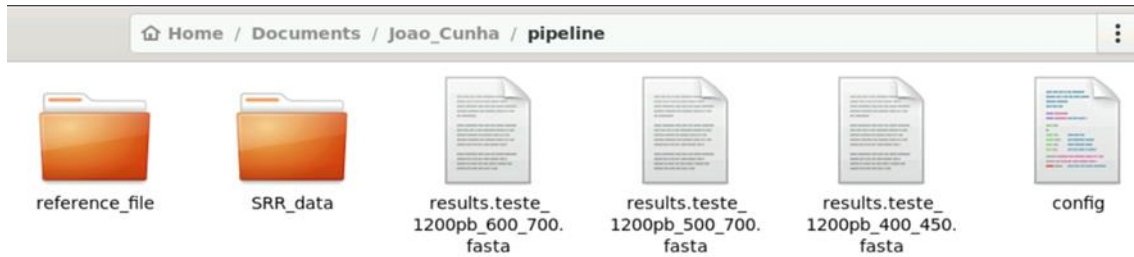


Figure 15) Example of the folder with simulated output files.

4.9.3. Usage with Real Sequences

It's important to note that when using real sequencing data, the pipeline's outcomes may not always be as straightforward as those obtained in controlled test sequences. Various factors, such as the complexity of real-world genomic data, differences in sequencing technologies, or the presence of variations and indels, can influence the results. Therefore, while the pipeline provides valuable insights into inversion breakpoints in authentic datasets, the interpretations may require a deeper understanding of the datasets and context. To illustrate the nuances of working with authentic data, the Figure 16 present an example of real sequencing data.

```

1 >INV_a-u_16_rep_6 Not found in reference genome
2 ACCGAGTACGAGTACGGGACAGTACGGGACAGAGTACGAGACGGAGTACGGGA
3 >INV_v-b_16_rep_6 NW_001845359.1 100.000 54 1011 1064 Found in reference genome
4 GGTCCCGCACTCGGTTCGTAATGATCCCGTTTTGGTCCCGTACTGGTCCTGTAC
5 >INV_a-vbar_16_rep_6 Not found in reference genome
6 ACCGAGTACGAGTACGGGACAGTACGGGATCAGTACGAACCGAGTGCGGGACC
7 >INV_ubar-b_16_rep_6 NW_001845359.1 98.148 54 722 775 INV_ubar-b_16_rep_6 NT_033777.3
  INV_ubar-b_16_rep_6 NT_033777.3 98.148 54 239599 239652 Found in reference genome
8 TCCCGTACTCCGTCTCGTACTCTGTCCCGTTTTGGTCCCGTACTGGTCCTGTAC
9 >INV_a-u_17_rep_8 Not found in reference genome
10 ACGGGACCTGTACAGGACCAGCACGGGACAGTACGGGACAGTACGGGGGC
11 >INV_v-b_17_rep_8 NW_001844948.1 98.077 52 11020 10969 INV_v-b_17_rep_8 NW_001844948.1
  INV_v-b_17_rep_8 NW_001846183.1 98.077 52 378 429 Found in reference genome
12 GGTCCCAAGCTGGTCTGTACAGGTCCCGTACTCGTTTCCGTACTGGTCCCG
13 >INV_a-vbar_17_rep_8 NW_001845040.1 100.000 52 3701 3650 Found in reference genome
14 ACGGGACCTGTACAGGACCAGCACGGGACCTGTACAGGACCAGCTTGGGACC
15 >INV_ubar-b_17_rep_8 Not found in reference genome
16 GCCCCGTACTGGTCCCGTACTGGTCCCGTACTCGTTTCCGTACTGGTCCCG
  
```

Figure 16) Example of Real Data Results file.

4.10. Troubleshooting

While the pipeline is designed to streamline the process of identifying inversion breakpoints, users may encounter common issues when working with diverse datasets and configurations. This section aims to address some of these challenges and provide guidance on how to resolve them effectively.

4.10.1. Common Issues and Solutions

Low Alignment Quality: If the pipeline's alignment quality appears to be suboptimal, it may be due to variations in sequencing depths, sequencing errors, or variations in the dataset. In such cases, consider the following steps:

Increase sequencing coverage: A higher sequencing coverage can enhance alignment quality and confidence in breakpoint detection.

Adjust parameters: Experiment with alignment and filtering parameters, such as read length, mutation rate, and error rate, to optimize results for your specific dataset.

Check data quality: Ensure that your input sequences are of high quality and free from artifacts or contamination.

Ambiguous Results: In some instances, the pipeline may produce ambiguous or inconclusive results. This can occur when breakpoints are located in repetitive regions of the genome or when dealing with complex structural variations. Here's how to approach this:

Validate results: Utilize additional validation techniques or alternative software tools to corroborate the identified breakpoints.

Investigate repetitive regions: Evaluate whether the identified breakpoints fall within repetitive or challenging genomic regions, as this can affect alignment accuracy. Use a lower value of genomic hits.

Resource Limitations: The pipeline's computational demands can vary depending on the dataset size and complexity. Resource limitations may lead to slow processing or errors.

4.10.2. Solutions

Optimize hardware: Ensure that your computational resources, such as RAM and CPU, are sufficient for the dataset size. Consider parallel processing if available.

Reduce dataset size: If feasible, sub-sample or partition large datasets to manage resource constraints.

Incomplete Breakpoint Identification: In some cases, the pipeline may not identify all breakpoints accurately, particularly when dealing with complex rearrangements.

Review data quality: Thoroughly inspect the quality and characteristics of your input data, as low-quality or incomplete data can hinder accurate breakpoint identification.

Utilize complementary tools: Combine results from multiple structural variation detection tools or conduct manual inspections to identify challenging breakpoints.

5. Conclusions and Further Perspectives

Using data from next-generation sequencing, the *Drosophila* Inversion Breakpoints Workflow has demonstrated a potent and automated method for precisely locating inversion breakpoints inside *Drosophila* genomes. By using freely accessible databases, combining specialist tools, and providing adjustable choices to accommodate users with different degrees of competence, this specialized bioinformatics pipeline accelerates the analysis process.

The key takeaways from this work include:

Streamlined Analysis: The *Drosophila* Inversion Breakpoints Workflow has significantly reduced the time and resources required for complex inversion breakpoint analysis. Its capability to function without a reference genome streamlines the process, making it efficient and accessible.

Database Reliance: The workflow's reliance on reputable databases like the European Nucleotide Archive (ENA) and the National Center for Biotechnology Information (NCBI) ensures the accuracy and accessibility of the data, establishing a strong foundation for research.

Specialized Tools: Integration of specialized bioinformatics tools, including Wgsim, AfterQC, TakeaBreak, and Blastn, plays a pivotal role. These tools are chosen for their specific functions in read simulation, quality assurance, breakpoint discovery, and validation, ensuring the robustness of the workflow.

Customization: To accommodate users with varying degrees of competence, the pipeline provides default settings for simplicity of use while allowing sophisticated users to modify parameters to meet their particular research goals.

Validation and Filtering: The addition of the Blastn validation step enhances the accuracy of identified breakpoints. Filtering criteria further refine results, ensuring that only the most relevant candidates proceed for further investigation.

Looking ahead, several future perspectives can further enhance the pipeline:

Computational Efficiency: Efforts will focus on optimizing computational efficiency. Techniques like parallel processing and resource allocation will be explored to handle

the increasing complexity of Next-Generation Sequencing (NGS) datasets, ensuring swift and effective analyses.

Long-Read Sequencing Integration: Long-read sequencing technologies will be used to increase the analysis's capacity for detecting complicated genomic rearrangements.

Structural Variant Detection: Integration with structural variant (SV) detection tools will allow for the simultaneous identification of various genomic rearrangements in a single analysis.

User Accessibility: Development of a user-friendly graphical interface is planned to make the pipeline more accessible to researchers with limited bioinformatics expertise.

Continuous Benchmarking: Ongoing efforts will be dedicated to continuous validation and benchmarking against a variety of datasets, including real-world data. This rigorous testing ensures the workflow's accuracy and reliability across diverse genomic scenarios, instilling confidence in its results.

In conclusion, the *Drosophila* Inversion Breakpoints Workflow stands as a pivotal tool in genomics research. Its automation, accessibility, and adaptability cater to the evolving landscape of genomics. By providing a potent solution for precise inversion breakpoint identification, the workflow accelerates research processes, opening avenues for deeper genomic exploration. As advancements continue, the workflow remains committed to enhancing accuracy, efficiency, and user-friendliness, contributing significantly to the field of genomics.

References

1. Kirkpatrick, M., *How and why chromosome inversions evolve*. PLoS biology, 2010. **8**(9): p. e1000501.
2. Sturtevant, A.H., *A case of rearrangement of genes in Drosophila*. Proceedings of the National Academy of Sciences, 1921. **7**(8): p. 235-237.
3. Puerma, E., D.J. Orengo, and M. Aguadé, *The origin of chromosomal inversions as a source of segmental duplications in the Sophophora subgenus of Drosophila*. Scientific Reports, 2016. **6**(1): p. 30715.
4. Castermans, D., et al., *Identification and characterization of the TRIP8 and REEP3 genes on chromosome 10q21.3 as novel candidate genes for autism*. European Journal of Human Genetics, 2007. **15**(4): p. 422-431.
5. Reis, M., et al., *Origin and consequences of chromosomal inversions in the virilis group of Drosophila*. Genome Biology and Evolution, 2018. **10**(12): p. 3152-3166.
6. Vieira, J., et al., *Discordant rates of chromosome evolution in the Drosophila virilis species group*. Genetics, 1997. **147**(1): p. 223-230.
7. Papaceit, M., M. Aguadé, and C. Segarra, *Chromosomal evolution of elements B and C in the Sophophora subgenus of Drosophila: evolutionary rate and polymorphism*. Evolution, 2006. **60**(4): p. 768-781.
8. Sonoda, E., et al., *Differential usage of non-homologous end-joining and homologous recombination in double strand break repair*. DNA repair, 2006. **5**(9-10): p. 1021-1029.
9. Ranz, J.M., et al., *Principles of genome evolution in the Drosophila melanogaster species group*. PLoS biology, 2007. **5**(6): p. e152.
10. Delprat, A., et al., *The transposon Galileo generates natural chromosomal inversions in Drosophila by ectopic recombination*. PLoS One, 2009. **4**(11): p. e7883.
11. Fonseca, N.A., et al., *The DAIBAM MITE element is involved in the origin of one fixed and two polymorphic Drosophila virilis phylad inversions*. Fly, 2012. **6**(2): p. 71-74.
12. Rius, N., A. Delprat, and A. Ruiz, *A divergent P element and its associated MITE, BuT5, generate chromosomal inversions and are widespread within the Drosophila repleta species group*. Genome biology and evolution, 2013. **5**(6): p. 1127-1141.

13. Richards, S., et al., *Comparative genome sequencing of *Drosophila pseudoobscura*: chromosomal, gene, and cis-element evolution*. *Genome research*, 2005. **15**(1): p. 1-18.
14. Cáceres, M., et al., *Generation of a widespread *Drosophila* inversion by a transposable element*. *Science*, 1999. **285**(5426): p. 415-418.
15. Dias, G.B., et al., *Tetris is a foldback transposon that provided the building blocks for an emerging satellite DNA of *Drosophila virilis**. *Genome biology and evolution*, 2014. **6**(6): p. 1302-1313.
16. Casals, F., M. Cáceres, and A. Ruiz, *The foldback-like transposon Galileo is involved in the generation of two different natural chromosomal inversions of *Drosophila buzzatii**. *Molecular biology and evolution*, 2003. **20**(5): p. 674-685.
17. Evans, A.L., P.A. Mena, and B.F. McAllister, *Positive selection near an inversion breakpoint on the neo-X chromosome of *Drosophila americana**. *Genetics*, 2007. **177**(3): p. 1303-1319.
18. Petrov, D.A. and D.L. Hartl, *High rate of DNA loss in the *Drosophila melanogaster* and *Drosophila virilis* species groups*. *Molecular Biology and Evolution*, 1998. **15**(3): p. 293-302.
19. Petrov, D.A., *DNA loss and evolution of genome size in *Drosophila**. *Genetica*, 2002. **115**: p. 81-91.
20. Capy, P., et al., *Dynamics and evolution of transposable elements*. 1998: Springer.
21. Feschotte, C. and E.J. Pritham, *DNA transposons and the evolution of eukaryotic genomes*. *Annu. Rev. Genet.*, 2007. **41**: p. 331-368.
22. Haren, L., B. Ton-Hoang, and M. Chandler, *Integrating DNA: transposases and retroviral integrases*. *Annual Reviews in Microbiology*, 1999. **53**(1): p. 245-281.
23. Tang, M., et al., *Guanosine triphosphate acts as a cofactor to promote assembly of initial *P*-element transposase–DNA synaptic complexes*. *Genes & development*, 2005. **19**(12): p. 1422-1425.
24. Hartl, D.L., A.R. Lohe, and E.R. Lozovskaya, *Modern thoughts on an ancyent marinere: function, evolution, regulation*. *Annual review of genetics*, 1997. **31**(1): p. 337-358.
25. McCullers, T.J. and M. Steiniger, *Transposable elements in *Drosophila**. *Mobile genetic elements*, 2017. **7**(3): p. 1-18.
26. Finnegan, D.J., *Eukaryotic transposable elements and genome evolution*. *Trends in genetics*, 1989. **5**: p. 103-107.

27. Morales-Hojas, R., C.P. Vieira, and J. Vieira, *The evolutionary history of the transposable element Penelope in the Drosophila virilis group of species*. Journal of molecular evolution, 2006. **63**: p. 262-273.
28. Petrov, D.A., et al., *Size matters: non-LTR retrotransposable elements and ectopic recombination in Drosophila*. Molecular biology and evolution, 2003. **20**(6): p. 880-892.
29. Behjati, S. and P.S. Tarpey, *What is next generation sequencing?* Archives of Disease in Childhood-Education and Practice, 2013. **98**(6): p. 236-238.
30. Rissman, A.I., et al., *Reordering contigs of draft genomes using the Mauve aligner*. Bioinformatics, 2009. **25**(16): p. 2071-2073.
31. Alkan, C., B.P. Coe, and E.E. Eichler, *Genome structural variation discovery and genotyping*. Nature reviews genetics, 2011. **12**(5): p. 363-376.
32. Medvedev, P., M. Stanciu, and M. Brudno, *Computational methods for discovering structural variation with next-generation sequencing*. Nature methods, 2009. **6**(Suppl 11): p. S13-S20.
33. Lemaitre, C., L. Ciortuz, and P. Peterlongo. *Mapping-free and assembly-free discovery of inversion breakpoints from raw NGS reads*. in *Algorithms for Computational Biology: First International Conference, AICoB 2014, Tarragona, Spain, July 1-3, 2014, Proceedigns 1*. 2014. Springer.
34. Li, H., et al., *The sequence alignment/map format and SAMtools*. bioinformatics, 2009. **25**(16): p. 2078-2079.
35. Milhaven, M. and S.P. Pfeifer, *Performance evaluation of six popular short-read simulators*. Heredity, 2023. **130**(2): p. 55-63.
36. Buffalo, V., *Bioinformatics data skills: Reproducible and robust research with open source tools*. 2015: " O'Reilly Media, Inc."
37. Jangla, K. and K. Jangla, *Docker Basics*. 2018: Springer.
38. López-Fernández, H., et al. *The pegi3s bioinformatics docker images project*. in *Practical Applications of Computational Biology & Bioinformatics, 15th International Conference (PACBB 2021)*. 2022. Springer.
39. Zhao, N., et al. *Large-scale analysis of the docker hub dataset*. in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. 2019. IEEE.
40. Liu, S. and Z. Liu, *Introduction to linux and command line tools for bioinformatics*. Bioinformatics in Aquaculture: Principles and Methods, 2017: p. 1-29.
41. Ramey, C. *Bash, the Bourne– Again Shell*. in *Proceedings of The Romanian Open Systems Conference & Exhibition (ROSE 1994), The Romanian UNIX User's Group (GURU)*. 1994.

42. Robbins, A., *Bash Pocket Reference: Help for Power Users and Sys Admins*. 2016: " O'Reilly Media, Inc."
43. Sayers, E.W., et al., *Database resources of the National Center for Biotechnology Information in 2023*. *Nucleic acids research*, 2023. **51**(D1): p. D29-D38.
44. Ogasawara, O., et al., *DDBJ Database updates and computational infrastructure enhancement*. *Nucleic acids research*, 2020. **48**(D1): p. D45-D50.
45. Burgin, J., et al., *The european nucleotide archive in 2022*. *Nucleic Acids Research*, 2023. **51**(D1): p. D121-D125.
46. Leinonen, R., et al., *Improvements to services at the European Nucleotide Archive*. *Nucleic acids research*, 2010. **38**(suppl_1): p. D39-D45.
47. Kodama, Y., M. Shumway, and R. Leinonen, *The Sequence Read Archive: explosive growth of sequencing data*. *Nucleic acids research*, 2012. **40**(D1): p. D54-D56.
48. Sayers, E.W., et al., *GenBank*. *Nucleic acids research*, 2022. **50**(D1): p. D161-D164.
49. Ciufu, S., et al., *Using average nucleotide identity to improve taxonomic assignments in prokaryotic genomes at the NCBI*. *International journal of systematic and evolutionary microbiology*, 2018. **68**(7): p. 2386.
50. Kitts, P.A., et al., *Assembly: a resource for assembled genomes at NCBI*. *Nucleic acids research*, 2016. **44**(D1): p. D73-D80.
51. O'Leary, N.A., et al., *Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation*. *Nucleic acids research*, 2016. **44**(D1): p. D733-D745.
52. Chen, S., et al., *AfterQC: automatic filtering, trimming, error removing and quality control for fastq data*. *BMC bioinformatics*, 2017. **18**(3): p. 91-100.
53. Altschul, S.F., et al., *Basic local alignment search tool*. *Journal of molecular biology*, 1990. **215**(3): p. 403-410.
54. Li, H., *wgsim-Read simulator for next generation sequencing*. Github repository, 2011.
55. Ramirez-Gonzalez, R.H., et al., *Bio-samtools: Ruby bindings for SAMtools, a library for accessing BAM files containing high-throughput sequence alignments*. *Source code for biology and medicine*, 2012. **7**(1): p. 1-6.
56. Hu, T., et al., *Next-generation sequencing technologies: An overview*. *Human Immunology*, 2021. **82**(11): p. 801-811.