Plateforme de bioinformatique ARTbio

User Manual & Tutorial

SRAX

Systematic Resistome Analysis Tool

Leonardo G. Panunzi

version: 7th March 2019

Contents

1 Abbreviations

Abbreviation	**Phrase**
NGS	Next-generation sequencing
CNV	copy number variation
SV	structural variation
OS	Operative System

2 Introduction

2.1 AMR context

Microbial drug resistance is a global healthcare problem caused by the extensive uncontrolled use of antibiotics in medicine and agriculture. It is predicted that toward 2050 around 10 million people will die annually for reasons connected with drug resistance (O'Neill,2016). Although resistant pathogens are the main concern, the global microbial channels of gene exchange existing between unrelated microbial taxa allow commensal microbes to share resistance genes with opportunists.

Genomic information has been incorporated in order to under- stand why certain strains of pathogens are resistant to antibiotics, including Staphylococcus aureus, Mycobacterium tuberculosis, Klebsiella pneumoniae, Salmonella spp. and Pseudomonas aerugi- nosa (Bradley et al., 2015; Gordon et al., 2014; McDermott et al., 2016; Stoesser et al., 2013; Tyson et al., 2015).

Antimicrobial resistance (AMR) remains a significant global threat to public health, with directly attributable deaths per annum pre- dicted to rise from 700 000 to 10 000 000 by the year 2050 (OâĂŹNeill, 2016). This threat is not restricted to humans, with consequences also for animal health, welfare and food production (Bengtsson and Greko, 2014). In an effort to inform policy that can mitigate the spread of AMR, there has been a recent drive to establish surveil- lance programmes to monitor the prevalence of AMR (Public Health Agency of Canada, 2016; World Health Organization, 2015). These programmes have traditionally used cul- ture or polymerase chain reactionâĂŞbased surveillance techniques, restricting monitoring to a few key genes or organisms. However, the use of metagenomics for surveilling AMR is gaining traction as it offers a much greater breadth of testing over these traditional techniques (Baquero, 2012; Miller et al., 2013). The use of metagenomics to determine the antibiotic resistance gene (ARG) content of a microbial community, hereafter referred to as resistome profiling, has been applied in studies of a wide variety of biomes (Auffret et al., 2017; Jalali et al., 2015; Ma et al., 2017; Munk et al., 2017; Rose et al., 2017; Rowe et al., 2017, 2016; Tao et al., 2016; Yang et al., 2013). Studies such as these utilize several well-maintained ARG databases (Gupta et al., 2014; Jia et al., 2017; Zankari et al., 2012) and ARG-annotation tools (Hunt et al., 2017; Inouye et al., 2014; Rowe et al., 2015; Yang et al., 2016), of which only a few are designed for resistome profiling (Fig. 1). These data- bases and tools are used with assembled metagenomic contigs, or al- ternatively with metagenomic sequence reads. In the case of the latter some tools and studies opt to align reads to reference ARGs and report only fully covered references (Rowe et al., 2017), where- as some bin reads using BLAST and similarity/length thresholds (Tao et al., 2016). All of these strategies offer the user a compromise between ease of use, analysis speed and the accurate typing of ARGs (ability to detect ARG type/subtype). One of the main limitations of existing tools for resistome profil- ing (or profiling of other gene sets) in metagenomes is that high simi- larity shared between reference sequences can result in ambiguous alignments, unaligned reads (false negatives), or mis-annotated reads (false positives); thus reducing the accuracy when typing ARGs (Petersen et al., 2017). A solution to this has been to cluster the ref- erence sequences and then use the cluster representative sequences as a target for read alignment (Rowe et al., 2015); however, this results in a loss of information as ARG subtypes will be masked. Likewise, a related approach has been to collapse final annotations of ARG variants into a common ARG annotation, also masking sub- types (Munk et al., 2017). The ability to accurately detect ARG type/subtype, here termed profiling resolution, is important consid- ering that the variation between subtypes of ARGs can result in different phenotypic activity (Bush and Jacoby, 2010).

In this article, we present a method for resistome profiling that utilizes a variation graph representation of ARG databases to reduce ambiguous alignment of metagenomic sequence reads. We store sets of similar ARG reference sequences in variation graphs; collapsing identical sequences whilst retaining unique nodes that allow for ac- curate typing. By applying an LSH Forest indexing strategy to the variation graph collection that allows for fast and approximate search queries, we show that metagenomic reads can be seeded against candidate graph traversals. Subsequent hierarchical local alignment of reads and scoring enables accurate and efficient resis- tome profiling. We also provide our implementation, Graphing Resistance Out Of meTagenomes (GROOT), and compare it against ARGS-OAP and AMRPlusPlus, the most recently published tools for resistome profiling from metagenomic data (Lakin et al., 2017; Yang et al., 2016) (Fig. 1).

Antimicrobial-resistant (AMR) pathogens greatly undermine people’s ability to control pathogens and cure diseases. The ultra-fast mutation rates of these microbes render our existing drugs useless against superbugs, and âĂŸexisting classes of antibiotics are probably the best we will ever have (Cormican and Vellinga, 2012).âĂŹ A study published in 2013 also identified that, due to AMR, additional eco- nomic costs may be as high as 55 billion USD and that trivial bacter- ial infections such as hip replacements may increase the death rate from approximately 0% to 30% (Smith and Coast, 2013). The rapid decrease in the number of new drugs further diminishes our chances

Here I present **sraX**, an interactive tool for comprehensive visualization of the gut resistome in populations of the world. The displayed features include the relative abundance of AR genes, AR-conferring mutations as well as genes conferring resistance to bio- cides and heavy metals. ResistoMap is a perspective tool for exploring the global landscape of gut resistome in order to identify national

(Panunzi and Agüero 2014).

2.2 Implementation

sraX constitutes a Perl package, composed of functional modules, that allows detecting the presence of AMR determinants and, ultimately, describing the repertoire of antibiotic resistance genes (ARGs) within a collection of genomes (the "resistome" analysis). While running a single-step command on a bash terminal, the proposed tool is capable of scanning assembled sequence files in FASTA format and systematically automating the following assignments:

- creation and compilation of a local AMR database (DB) using public or proprietary repositories. — there is a steep 'learning curve'.[a]
- Creation and compilation of a local AMR database (DB) using public or proprietary repositories.
- Accurate identification of AMR determinants (ARGs or SNPs presence) in a non-redundant manner
- Detection of putative new variants through the SNP analysis
- Calculation and graphical representation of drug classes and type of mutated loci
- In-depth gene context exploration

The results are presented in fully navigable HTML-formatted files with embedded plots of previously mentioned analysis.

3 Installation

3.1 Dependencies

1. Though **sraX** is fully written in Perl and should work with any OS, it has only been tested with a 64-bit Linux distribution. Finally, for very large amounts of information R can run out of memory because all the calculations are done in RAM.[b]

[a] A steep learning curve means you learn a lot per unit time …

[b] Now that there is a 64-bit version of R for Windows, this is less of a problem and the limitation is more to do with the amount of RAM the computer has.

2. Perl version 5.26.x or higher. You can verify on your own computer by typing the following command under a bash terminal:

   ```
   perl -h
   ```

   The latest version of Perl can be obtained from the official website. Consult the installation guide.

   The following Perl libraries are also required and can be installed using CPAN:

   - LWP::Simple
   - Data::Dumper
   - JSON
   - File::Slurp
   - FindBin
   - Cwd

3. Third-party software:

   - BLAST
   - DIAMOND
   - MUSCLE Edgar (2004)
   - R
   - ‘dplyr‘
   - ‘ggplot2‘
   - ‘gridExtra‘

   NOTE: The bash script “install_srax.sh” is provided, in order to confirm the existence of these dependencies in your computer. If any of them would be missing, the bash script will guide you for a proper installation.

3.2 Installation

To successfully install **sraX**, please see the details provided below. If you encounter an issue during the process, please contact your local system administrator. If you encounter a bug please log it here or email me at `mailto:lgpanunzi@gmail.com`.

Open a bash terminal and clone the repository:

```
git clone https://github.com/lgpdevtools/sraX.git
```

To verify the existence of required dependencies and ultimately install the perl modules composing **sraX**, inside the cloned repository run:

```
sudo bash install_srax.sh
```

4 Usage

sraX effectively operates as one-step application. It means that just a single command is required to obtain the totality of results and their visualization.

NOTE: For a detailed explanation and examples from real datasets, please follow the Tutorial.

Parameters:

-i|input <Mandatory: input genome directory> -o|output <Optional: name of output folder> -p|align_seq <Optional: sequence aligning algorithm (default: dblastx)> -e|eval <Optional: evalue cut-off to filter false positives (default: 1e-05)> -c|aln_cov <Optional: fraction of aligned query to the reference sequence (default: 60)> -id <Optional: sequence identity percentage cut-off to filter false positives (default: 85)> -t|threads <Optional: number of threads to use (default: 6)> -v|version <print current version> -d|debug <Optional: print verbose output for debugging (default: No)> -h|help <print this message>

`-i|input` The name of the directory [/path/to/input_dir] containing the input FASTA file(s).

`-o|output` The name of the directory where to store the results. Default: `-o 'input_sraX_id_c`

`-p|palign` The selected algorithm for aligning each genome file to the `AMR DB`. For **faster** processing `dblastx` (DIAMOND blastx) should be employed, while `blastx` (NCBI blastx) is recommended for **higher accuracy**. In any case, the process is parallelized (up to 100 genome files are run simultaneously) for reducing computing times. Default: `-p dblastx`

`-e|eval` Minimum evalue cut-off to filter false positives. Default: `-e 1e-05`

`-c|aln_cov`

`-id`

`-t|threads`

`-v|version`

```
-h|help
```

Minimal command Example usage:

```
sraX -i [/path/to/input_genome_directory]
```

Where:

```
-i  Full path to the mandatory directory containing the input
   sequence data, which must
    be in FASTA format and consisting of individual assembled
       genome sequences.
```

Extended options Example usage:

```
sraX -p blastx -id 95 -c 90 -t 12 -o [/path/to/
   output_results_directory] -i [/path/to/input_genome_directory
   ]
```

Where:

```

-p  The translated alignments of assembled genome(s) are queried
   using dblastx
    (DIAMOND blastx) or blastx (NCBI blastx). In any case, the
       process is parallelized
    (up to 100 genome files are run simultaneously) for reducing
       computing times
    (default: dblastx)

-id Use this percent identity cut-off to filter false positives
   (default: 85)

-c  Use this fraction aligned query to the reference sequence (
   default: 60)

-e  Use this evalue cut-off to filter false positives (default: 1
   e-05)

-t  Use this number of threads (default: 6)

-o  Full path to the directory where the output results will be
   written in. If not provided,
    the following default name will be taken:

```

```
    'genome\_directory'\_'sraX'\_'id'\_'aln\_cov'\_'align\_seq'

    Example:
    input directory = 'Test'; -id 85; -c 95; -p dblastx

    Output directory = 'Test\_sraX\_85\_95\_dblastx'

-i  Full path to the mandatory directory containing the input
   sequence data, which must
    be in FASTA format and consisting of individual assembled
       genome sequences.
```

4.1 Creating a custom AMR DB

1) Increasing the AMR DB volume and its detection range User-provided data can be originated from public repositories or from its own.

Public ARG sequences repositories:

A) [ARGDIT](https://github.com/phglab/ARGDIT)

A recently published work [[1]](https://doi.org/10.1093/bioinformatics/bty987) describes the **ARGDIT** toolkit for creating curated AMR DBs. The authors provided already [pre-compiled AMR DBs](https://github.com/phglab/ARGDIT/tree/master/sample$_{i}ntegrated_{d}bs)asexample$ $*sraX**'spracticalityandutilityforresistomeprofiling.$

The curated ARG data will be downloaded and the headers will be formatted for being effective for **sraX** analysis. Using the bash console, run the following commands:

```
wget -O User_provided_ARGs/Public_repositories/argdit_dna.fa
   https://github.com/phglab/ARGDIT/blob/master/
   sample_integrated_dbs/argdit_nt_db.fa?raw=true

awk -F \| '/^>/ { print ">"$2"|"$1"|"$3"|protein_homolog|"$9"|"
   $5; next } 1' User_provided_ARGs/Public_repositories/
   argdit_dna.fa > User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa

sed -i 's/|>/|/g' User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa

rm -f User_provided_ARGs/Public_repositories/argdit_dna.fa
```

B) [NCBI Bacterial Antimicrobial Resistance Reference Gene Database](https://www.ncbi.nlm.nih.gov

The NCBI public repository is comprised by annotated DNA sequences that encode proteins conferring AMR, and it constitutes an aggregation of collections from multiple sources which were previously curated and further expanded by reviewing the dedicated literature.

Using the bash console, run the following commands:

```
wget -O User_provided_ARGs/Public_repositories/ncbi_aa.fa ftp://
   ftp.ncbi.nlm.nih.gov/pathogen/Antimicrobial_resistance/
   AMRFinder/data/latest/AMRProt

awk -F \| '/^>/ { print ">"$6"|"$2"|"$9"|protein_homolog|"$8"|
   Not_indicated"; next } 1' User_provided_ARGs/
   Public_repositories/ncbi_aa.fa > User_provided_ARGs/
   Public_repositories/ncbi_aa.fancbi_aa_formatted.fa

rm -f User_provided_ARGs/Public_repositories/ncbi_aa.fa
```

User's own ARG sequences:

2) Genome data acquisition from already analyzed collections

Data-set 1: 52 genomes belonging to $_Escherichiacoli_[$[2]]($https://doi.org/10.1093/jac/dkw511)The$

A) Go to the following **NCBI** repository and download all the genome assemblies: [[data-set-1]](https://www.ncbi.nlm.nih.gov/assembly?LinkName=bioproject$_assembly_all from_uid =$ 335932)

$_Note_The following steps are recurrent and should be followed for performing the **sraX** analysis with alternative genome data-sets. For analyzing othe$

B) Move the compressed downloaded file to the working directory and, using the bash console, extract the genome data and rename the directory:

```
mv /download_full_path/genome_assemblies.tar /
   working_dir_full_path/
  tar -zxf genome_assemblies.tar
  rm -f genome_assemblies.tar
  mv genome_assemblies ds1
```

C) Run **sraX** using the desired options. The following one is just an example using default options:

```
  sraX -d ds1
```

D) Adding user-provided ARG sequences:

```
sraX -d ds1 -u User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa
```

E) Modifying the amino-acid identity percentage and alignment coverage for detecting positive hits:

```
sraX -d ds1 -u User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa -i 95 -c 95
```

F) Modifying the output result directory:

```
sraX -d ds1 -u User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa -i 75 -c 90 -o ds1_another_test
```

Data-set 2: 76 genomes belonging to $_{E}scherichiacoli_{[}[3]](https://academic.oup.com/jac/article/70$
$resistant_{E}.coli_{f}romfarmisolatesandidentifiedthespecificgeneticdeterminantscontributingtoA$

A) Go to the following **NCBI** repository and download all the genome assemblies: [[data-set-2]](https://www.ncbi.nlm.nih.gov/assembly?LinkName=bioproject$_{a}ssembly_{a}ll from_{u}id =$ $266657)$

B) Follow previous procedures and only change the genome directory name:

```
mv genome_assemblies ds2
```

C) Run **sraX** using your own options. The following one is just an example:

```
sraX -d ds2 -u User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa
```

Data-set 3: 641 genomes belonging to $_{S}almonellaentericaspp_{[}[4]](https://doi.org/10.1128/AAC.010$
$16)$

The authors look at the phenotype and genotype correlation in $_{S}almonellaenterica_{w}ithdifferentantib$

A) Go to the following **NCBI** repository and download all the genome assemblies: [[data-set-3]](https://www.ncbi.nlm.nih.gov/assembly?LinkName=bioproject$_{a}ssembly_{a}ll from_{u}id =$ $242614)$

B) Follow previous procedures and only change the genome directory name:

```
mv genome_assemblies ds3
```

C) Run **sraX** using your own options. The following one is just an example:

```
sraX -d ds3 -u User_provided_ARGs/Public_repositories/
   argdit_dna_formatted.fa
```

Clinical context

`Jupyter` and `Uranus` were diagnosed with Acute Lymphoblastic Leukemia B (ALL-B). Bone marrow samples were used to prepare DNA from blastic cells, and this DNA was further enriched for sequences corresponding to 2 regions of chromosome 9 and chromosome 22, respectively, using capture probes. Captured DNA was then subjected to illumina paired-end sequencing.

5 Computational Analysis

The availability of enhanced analytical algorithms embedded in dedicated software allow the discovery and genotyping of intricated SVs. In the present document I will explain the required procedure for analyzing NGS data and evincing genomic translocations (or even further SNP detection) by running a series of bash commands in the terminal of a computer under a Linux OS.

6 Define the working environment

```
mkdir -p ART_bio_analysis/RefSeq
mkdir ART_bio_analysis/Seq_Reads
mkdir ART_bio_analysis/Results
cd ART_bio_analysis
```

7 Download the required sequence data

7.1 Download the patients' sequenced DNA

```
wget https://storage.googleapis.com/artbio_genomic_analysis/
   Jupyter_R1.fastq.gz -O ./Seq_Reads/Jupyter_R1.fastq.gz
wget https://storage.googleapis.com/artbio_genomic_analysis/
   Jupyter_R2.fastq.gz -O ./Seq_Reads/Jupyter_R2.fastq.gz
wget https://storage.googleapis.com/artbio_genomic_analysis/
   Uranus_R1.fastq.gz -O ./Seq_Reads/Uranus_R1.fastq.gz
wget https://storage.googleapis.com/artbio_genomic_analysis/
   Uranus_R2.fastq.gz -O ./Seq_Reads/Uranus_R2.fastq.gz
```

7.2 Download the latest assembly of the Human reference genome

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/
   GCF_000001405.38_GRCh38.p12/GCF_000001405.38_GRCh38.
   p12_genomic.fna.gz -O ./RefSeq/hg38.fna.gz
```

7.3 Download the annotation of the Human reference genome

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/
   GCF_000001405.38_GRCh38.p12/GCF_000001405.38_GRCh38.
   p12_genomic.gff.gz -O ./RefSeq/hg38.gff.gz
```

7.4 Uncompress the Human reference genome downloaded data

```
gunzip ./RefSeq/*.gz
```

8 Process sequence data

8.1 Construct the FM-Index

```
bwa index RefSeq/hg38.fna
```

8.2 Alignment of sequence reads to the Human reference genome

```
bwa mem -t 2 -R '@RG\tID:id\tSM:sample\tLB:lib' RefSeq/hg38.fna
   Seq_Reads/Jupyter_R1.fastq.gz Seq_Reads/Jupyter_R2.fastq.gz |
    samblaster --excludeDups --addMateTags --maxSplitCount 2 --
   minNonOverlap 20 | samtools view -S -b - > Results/
   Jupyter_aln.bam
bwa mem -t 2 -R '@RG\tID:id\tSM:sample\tLB:lib' RefSeq/hg38.fna
   Seq_Reads/Uranus_R1.fastq.gz Seq_Reads/Uranus_R2.fastq.gz |
   samblaster --excludeDups --addMateTags --maxSplitCount 2 --
   minNonOverlap 20 | samtools view -S -b - > Results/Uranus_aln.
   bam
```

8.3 Get some basic statistical parameters

```
samtools flagstat Results/Uranus_aln.bam
samtools flagstat Results/Jupyter_aln.bam
```

8.4 Selection of discordant paired-end alignments

```
samtools view -b -F 1294 Results/Jupyter_aln.bam > Results/
   Jupyter_disc_unsorted.bam
samtools view -b -F 1294 Results/Uranus_aln.bam > Results/
   Uranus_disc_unsorted.bam
```

8.5 Selection of split-read alignments

```
samtools view -h Results/Jupyter_aln.bam
   extractSplitReads_BwaMem -i stdin | samtools view -Sb - >
   Results/Jupyter_split_unsorted.bam
samtools view -h Results/Uranus_aln.bam extractSplitReads_BwaMem
    -i stdin | samtools view -Sb - > Results/
   Uranus_split_unsorted.bam
```

8.6 Sorting of the alignments

```
samtools sort Results/Jupyter_disc_unsorted.bam Results/
   Jupyter_discordants.bam
samtools sort Results/Jupyter_split_unsorted.bam Results/
   Jupyter_splitters.bam
samtools sort Results/Uranus_disc_unsorted.bam Results/
   Uranus_discordants.bam
samtools sort Results/Uranus_split_unsorted.bam Results/
   Uranus_splitters.bam
```

8.7 Index the alignment files

```
Results/Jupyter_aln.bam
samtools index Results/Jupyter_discordants.bam
samtools index Results/Jupyter_splitters.bam
Results/Uranus_aln.bam
samtools index Results/Uranus_discordants.bam
samtools index Results/Uranus_splitters.bam
```

9 Structural variant discovery

Now that we have an alignment of our sequence data against the Human reference genome, we are going to use **`LUMPY`**, that integrates multiple SV signals jointly across multiple samples.

9.1 Lumpy Express approach

Now that we have an alignment of our sequence data against the Human reference genome, we are going to use **`LUMPY`**, that integrates multiple SV signals jointly across multiple samples.

```
lumpyexpress -B Results/Jupyter_aln.bam -S Results/
   Jupyter_splitters.bam -D Results/Jupyter_discordants.bam -o
   Results/Jupyter_lumpy.vcf
lumpyexpress -B Results/Uranus_aln.bam -S Results/
   Uranus_splitters.bam -D Results/Uranus_discordants.bam -o
   Results/Uranus_lumpy.vcf
```

9.2 TIDDIT approach

`TIDDIT` allows the identification of intra and inter-chromosomal translocations, deletions, tandem-duplications, intersperesed duplications and inversions, using supplementary alignments as well as discordant pairs.

```
TIDDIT --sv -ploidy 2 -b Results/Jupyter_aln.bam -o Results/
   Jupyter_tiddit.vcf
TIDDIT --sv -ploidy 2 -b Results/Uranus_aln.bam -o Results/
   Uranus_tiddit.vcf
```

9.3 Delly approach

`Delly` constitutes a SV prediction method for discovering, genotyping and visualizing deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data. It uses paired-ends, split-reads and read-depth to sensitively and accurately delineate genomic rearrangements throughout the genome.

9.4 Indexing the Human reference genome & BCF format output

```
samtools faidx RefSeq/hg38.fna
delly call -x RefSeq/hg38.excl -o Results/Jupyter_delly.bcf -g
   RefSeq/hg38.fna Results/Jupyter_aln.bam
bcftools view Results/Jupyter_delly.bcf > Results/Jupyter_delly.
   vcf
```

```
delly call -x RefSeq/hg38.excl -o Results/Uranus_delly.bcf -g
   RefSeq/hg38.fna Results/Uranus_aln.bam
bcftools view Results/Uranus_delly.bcf > Results/Uranus_delly.
   vcf
```

9.5 Genotyping SVs

The genotypes detected on **LUMPY** or **TIDDIT** output VCF files can be called using a Bayesian maximum likelihood algorithm. **SVTyper** performs breakpoint genotyping of SVs directly from VCF files by assessing discordant and concordant reads from paired-end and split-read alignments to infer genotypes at each site.

```
svtyper -B Results/Jupyter_aln.bam -S Results/Jupyter_splitters.
   bam -i Results/Jupyter_lumpy.vcf > Results/Jupyter_svtyped.
   vcf
svtyper -B Results/Uranus_aln.bam -S Results/Uranus_splitters.
   bam -i Results/Uranus_lumpy.vcf > Results/Uranus_svtyped.vcf
```

Conclusion Chromosomal translocation in leukemia has been recognized as a genetic abnormality related to cancer progression long time ago. These specific rearrangements are likely the founding events during leukemogenesis. While focal sequence mutations can evolve during ALL progression and relapse, gene fusions arising from translocations usually remain unchanged across subclones over time, suggesting its fundamental role in the pathogenesis of this type of cancer. These chromosomal aberrations, along with the putative occurrence of missense mutations, are characteristic leukemogenic alterations that must be properly identified and characterized.

This document seeks to establish an standarized method for detecting genomic rearrangements in B-ALL analyzed genomes. Three different *in silico* approaches, with previously proven efficacy, are proposed for performing this task and complementing the diagnostic laboratory in bone marrow samples from leukemia patients.

References

Edgar, R. C. (2004). "MUSCLE: a multiple sequence alignment method with reduced time and space complexity". In: *BMC bioinformatics* 5.1, p. 113 (cit. on p. 9).

Panunzi, L. G. and F. Agüero (2014). "A genome-wide analysis of genetic diversity in Trypanosoma cruzi intergenic regions". In: *PLoS neglected tropical diseases* 8.5, e2839 (cit. on p. 8).