RECyT
Year 24 / Nº 38 / 2022 / 63–70
DOI: https://doi.org/10.36995/j.recyt.2022.38.008

Computational Environment for Genomic Sequencing and Annotation: a workflow for application in projects by life researchers

Configuración de Ambiente Computacional para el Ensamblaje y Anotación de Genomas: orientaciones para investigadores del Ciencia de la Vida

Configuração de Ambiente Computacional para Montagem e Anotação Genômica: orientações para pesquisadores da Ciência da Vida

Matheus Pedron Cassol[1, *], Alexandre Rafael Lenz[1,2], Rudinei Zacaria[1], Scheila de Avila e Silva[1]

1- Universidade de Caxias do Sul, Brasil; 2- Universidade do Estado da Bahia, Brasil.

* E-mail: mpcassol@ucs.br

Received: 01/09/2021; Accepted: 21/03/2022

Abstract

The current paper seeks to approach, using a workflow, basics subjects of the bioinformatic field and also useful informations to consider during the development of *in silico* researches. Installation and general usage of multiple softwares related to different sections of the genome annotation process were also presented. At last, an model organism, *Staphylococcus aureus,* was sequenced in two different softwares, SPAdes and IDBA-UD, seeking further comparison and evaluation of the process as a whole. The quality evaluation of the assemble was established by tests on QUAST, BUSCO and Augustus, supported by BLASTP. Results: QUAST evaluation returned genome coverage values above 98% in both test cases, pointing towards a trustworthy assemble for this organism. Via SPAdes were needed less computational resources, but, using IDBA-UD the sequences found were more contiguous. Results deriving from BUSCO showed only one expected gene difference. Some proteins and genes predicted by Augustus led to hits, sequences already studied in that organism, using the BLASTP program.

Keywords: Bioinformatic; *workflow*; assemble; genome; computational tools.

Resumen

El presente trabajo trata sobre un enfoque en formato de flujo de trabajo de cuestiones básicas del área, así como información a tener en cuenta durante la elaboración de investigaciones in *silico*. Se centró en algunos programas de diferentes partes del proceso de ensamblaje genómico, proporcionando orientación sobre su instalación y uso. Finalmente, se secuenció un organismo modelo, *Staphylococcus aureus*, en dos softwares, SPAdes e IDBA-UD, para la comparación y evaluación cualitativa del resultado. La evaluación de la calidad de la secuenciación se estableció mediante pruebas en los programas QUAST, BUSCO y Augustus, con el apoyo de BLASTP. La evaluación a través de QUAST arrojó valores de integridad en relación con el genoma de referencia superiores al 98% para ambas pruebas, lo que indica un ensamblaje confiable para el organismo en cuestión. La herramienta SPAdes logró secuenciar con menor capacidad computacional, pero a través de IDBA-UD se obtuvieron secuencias más contiguas. Los resultados de BUSCO mostraron solo una diferencia genética esperada. Las proteínas y genes esperados obtenidos por Augustus provocaron aciertos a través de BLASTP, es decir, secuencias de proteínas ya estudiadas y descritas para el organismo.

Palabras clave: Bioinformática; *workflow*; ensamblaje; genoma; herramientas computacionales.

Resumo

O presente trabalho trata-se de uma abordagem em formato de *workflow* de questões base da área de bioinformática, assim como informações para se levar em consideração durante a elaboração de pesquisas *in silico*. Focou-se em alguns programas de diferentes partes do processo de montagem genômica, fornecendo

orientações acerca de sua instalação e uso. Por fim, sequenciou-se um organismo modelo, *Staphylococcus aureus*, em dois softwares, SPAdes e IDBA-UD, para fins de comparação e avaliação qualitativa do resultado. A avaliação da qualidade do sequenciamento foi estabelecida por testes nos programas QUAST, BUSCO e pelo Augustus, apoiado pelo BLASTP. a avaliação via QUAST retornou valores de completude em relação ao genoma referência acima de 98% para ambos testes, indicando uma montagem confiável para o organismo em questão. Via SPAdes foi-se capaz de sequenciar com menor capacidade computacional, porém por intermédio do IDBA-UD obteve-se sequências mais contíguas. Os resultados advindos do BUSCO apresentaram apenas um gene esperado de diferença. As proteínas e genes esperados obtidos pelo Augustus suscitaram *hits* via BLASTP, ou seja, sequências proteicas já estudadas e descritas para o organismo.

Palavras-chave: Bioinformática; *workflow*; montagem; genoma; ferramentas computacionais.

Introduction

Bioinformatics is proving to be of great value for data manipulation in the field of biology. This area of knowledge encompasses all aspects related to the acquisition, processing, storage, distribution, analysis and interpretation of biological information. By unifying concepts and techniques from mathematics, statistics and computer science, it can lead to tools capable of extending the understanding of possible biological implications arising from genomic data [1].

Among the applications of bioinformatics, we find the sequencing and annotation of genomes, being the first responsible for providing the nucleotide composition of the genome of an organism. This technique is carried out by equipment from Illumina[1], Ion Torrent[2], Pacific Biosciences (PacBio)[3] and Oxford Nanopore Technology (ONT)[4], for example. This equipment allows the reading of text files containing DNA fragments called reads, which must be organised in order to represent the organism's genome. This procedure is called genome assembly, the most common approach being de novo, in which the genome is reconstructed exclusively from the overlay information of the reads. The sequences resulting from genome assembly are called contigs. For prokaryotic organisms, genome assembly can be performed by considering another genome as a reference model to guide the mapping of reads or by rearranging the contigs resulting from previous assembly [2,3]. Finally, the contigs are organised to form scaffolds, which will be used to perform genome annotation. Some examples of software used in this initial stage are: Velvet [4], SPAdes [5] and Trinity [6].

After sequencing, it is possible to obtain structural and functional information about the genome under investigation, as well as data that contribute to the evolutionary knowledge of organisms, innovations in diagnostic methods, new drugs, vaccines, possible means of prevention and more effective treatments against diseases or pests, and many other applications. Such information is obtained through annotation, which can be understood as a multilevel computational process involving nucleotides, proteins and processes [7].

In this context, for a project dedicated to genome assembly and annotation to develop successfully, it is necessary to initially define the application platform and the operating system to be used. After acquiring the equipment, translating the demands of a project into the specifications of a server (hardware) is fundamental to dimension the current needs. Depending on the chosen applications, it is important to carry out an analysis of the processing volume, memory usage and disk space consumption. For assembly, execution times and memory requirements increase with the amount of data. Therefore, there is a positive correlation between genome size and execution time and memory requirements [3]. For example, according to the company DNA STAR, which specialises in genomic sequencing, the organism Saccharomyces cerevisiae has a genome of approximately 15 MBases, thus requiring approximately 20 Gb of RAM, with 1 GB of RAM per MBase of genome length being recommended. These values can also be changed depending on the sequencer chosen and its mode of operation.

Genome assembly and annotation are purely computational procedures, usually performed by software without a graphical interface in a UNIX environment, which can be challenging for beginners in this environment. In addition, the results produced by one tool are not always in a format that can be used in the next tool in a workflow. In the literatura, it is possible to find articles that report the steps and procedures necessary for the execution of genomic assembly and annotation, and also discuss the

1- ILLUMINA. Illumina: sequencing and array-based solutions for genetic research. Sequencing and array-based solutions for genetic research. Available in: <https://www.illumina.com/>. Accessed on: 01 jul. 2021.

2- THERMO FISHER SCIENTIFIC BR. Ion Torrent. Available in: <https://www.thermofisher.com>. Accessed on: 01 jul. 2021.

3- PACIFIC BIOSCIENCES. PacBio: sequence with confidence. Sequence with confidence. Available in: <https://www.pacb.com>. Accessed on: 01 jul. 2021.

4- OXFORD NANOPORE TECHNOLOGIES. Nanopore. Available in: <https://nanoporetech.com/>. Accessed on: 01 jul. 2021.

peculiarities involved for each type of organism. In this regard, Zhou et al. [8] present the appropriate tools for genome assembly according to the sequencing approach. Keith [9] presents a compilation of methodological aspects related to genomic sequencing, assembly, annotation, data management, protein analysis and phylogenetic analysis. In the same vein, Ekblom and Wolf [10] analyse the workflow of a genomics project, from experimental procedures in the laboratory to the resulting applications of genome projects. In addition, Del Ángel et al. [3] point out important aspects that need to be analysed at the different stages. However, no article addresses the computational-technical difficulties related to the software execution environment.

In this context, many researchers in the field of life are discouraged to perform purely in silico procedures, either by technical-computational aspects or by the beginning of Bioinformatics as a science in the country. Although the installation of the software seems a simple task, during the assembly and annotation project of the fungus Penicillium equinulattum, carried out by researchers from the University of Caxias do Sul, some peculiarities were noticed that are not described in the installation tutorials and that, despite having relatively simple solutions, require significant time to diagnose. Thus, this article addresses technical aspects related to the Information Technology Infrastructure for its application in genomic assembly and annotation projects by researchers in the field of life sciences.

Materials and Methods

The methodology consisted in the elaboration of an orientation workflow for the installation and configuration of a computer server dedicated to genomic assembly and annotation. In this regard, this section describes the software and hardware selection steps.

Genome Assembly and Annotation Tools

The choice of tools was based on the experience of the team involved in the P. echinulatum fungal assembly and annotation project, which includes researchers in the areas of Biology and Computer Science. In this sense, software or pipelines were sought to reduce computational difficulties, as shown in Table 1.

Table 1: Computational tools.

Tool	Target	Language	Reference
FastQC	Performs quality control of raw data from high-throughput sequencing pipelines.	Java	ANDREWS, Simon et al. FastQC: a quality control tool for high throughput sequence data. 2010. [11]
TrimGalore	Aims to automate the quality of the file containing the reads, with additional functionalities related to the quality control of the file containing the data to be sequenced.	Perl	KRUEGER, Felix. Trim galore. A wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files, v. 516, p. 517, 2015. [12]
KmerGenie	Practically and accurately analyses the best value of k to be used in sequencing, arriving at it using approximate abundance histograms over multiple possible values.	R e Python 2.7 ou maior	CHIKHI, R.; MEDVEDEV, P. Informed and automated k-mer size selection for genome assembly. Bioinformatics, [S.L.], v. 30, n. 1, p. 31-37, 3 jun. 2013. Oxford University Press (OUP). [13]
SOAP-denovo 2	Composed of six modules, this software works on correcting read errors, constructing de Bruijn graphs, joining contigs, mapping paired-end reads, constructing scaffolds, and filling spaces.	C e C++	LUO, Ruibang *et al*. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. Gigascience, [S.L.], v. 1, n. 1, p. 1-6, dez. 2012. Oxford University Press (OUP). [14]
Abyss 2.0	The successor assembly program to Abyss, which works on assembling genomes across large datasets using Bloom filters. It uses less memory and has competitive results with other assembly software.	C++	JACKMAN, Shaun D. *et al.* ABySS 2.0: resource-efficient assembly of large genomes using a bloom filter. Genome Research, [S.L.], v. 27, n. 5, p. 768-777, 23 fev. 2017. Cold Spring Harbor Laboratory. [15]
Velvet	A set of algorithms that act on genome assembly through Brujin graphs, eliminating errors and circumventing repeated sequences. It seeks greater efficiency in extremely short reads, from 25 to 50 base pairs, in the case of pairwise reads.	C	ZERBINO, Daniel R.. Using the Velvet de novo Assembler for Short Read Sequencing Technologies. Current Protocols In Bioinformatics, [S.L.], v. 31, n. 1, p. 1-13, set. 2010. Wiley. [4]
Spades	A multi-pipelined assembler that works on Brujin graphs, using kmers only in initial assembly, followed by graph-theoretic operations. Its main focus is genomic assembly with data obtained from single cells, such as bacteria.	Python	BANKEVICH, Anton *et al.* SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. Journal Of Computational Biology, [S.L.], v. 19, n. 5, p. 455-477, maio 2012. Mary Ann Liebert Inc. [5]
IDBA-UD y variantes	Successor to IDBA that seeks to assemble genomes with large differences prior to read coverage. It employs iteration of k values, being able to reconstruct longer contigs more accurately. It focuses on reference-free assemblies, while the IDBA-Hybrid version addresses cases where there are similar genomes as reference. In the case of transcriptomes, the use of the IDBA-Tran variant is recommended.	C++	PENG, Y. *et al*. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics, [S.L.], v. 28, n. 11, p. 1420-1428, 11 abr. 2012. Oxford University Press (OUP). [16]
QUAST y variantes	Tool developed with the intention of evaluating and comparing genome assemblies by different platforms, with or without reference genome.	Python e Perl	MIKHEENKO, Alla; PRJIBELSKI, Andrey; SAVELIEV, Vladislav; ANTIPOV, Dmitry; GUREVICH, Alexey. Versatile genome assembly evaluation with QUAST-LG. Bioinformatics, [S.L.], v. 34, n. 13, p. 142-150, 27 jun. 2018. Oxford University Press (OUP). [17]
BUSCO	Software that seeks to quantify evaluatively the assembly of genomes taking into account the number of genes presented. It uses a database composed of orthologous genes from six main phylogenetic clades.	Python	MANNI, Mosè et al. BUSCO update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes. arXiv preprint arXiv:2106.11799, 2021. [18]
Trimmomatic	Flexible preprocessing tool that considers optimized read pairs for data obtained through Illumina NGS.	Java	BOLGER, Anthony M.; LOHSE, Marc; USADEL, Bjoern. Trimmomatic: a flexible trimmer for illumina sequence data. Bioinformatics, [S.L.], v. 30, n. 15, p. 2114-2120, 1 abr. 2014. Oxford University Press (OUP). [19]
Trinity	Package with three independent modules that again performs full transcriptome reconstruction. It is capable of acting on several samples from direct reads of RNA sequences, especially those where there is no reference genome.	C++, Java e Python	HAAS, Brian J et al. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nature Protocols, [S.L.], v. 8, n. 8, p. 1494-1512, 11 jul. 2013. Springer Science and Business Media LLC. [6]

Hisat2	A tool that aligns RNA and DNA sequences using a Ferragina Manzini index, mapping them to a population of the human genome or to a single reference genome.	C++, Python e Java	KIM, Daehwan *et al*. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nature Biotechnology, [S.L.], v. 37, n. 8, p. 907-915, ago. 2019. Springer Science and Business Media LLC. [20]
Augustus	Structural annotation software. Through a probabilistic model of a DNA sequence that has a gene structure, it defines the probable distribution in the given set, pointing out its coding regions. It uses different fonts, which can be provided by the user, contained in the program or external.	C++, Perl, Python	KELLER, Oliver; KOLLMAR, Martin; STANKE, Mario; WAACK, Stephan. A novel hybrid gene prediction method employing protein multiple sequence alignments. Bioinformatics, [S.L.], v. 27, n. 6, p. 757-763, 6 jan. 2011. Oxford University Press (OUP). [21]
GeneMark-ET	An ab-initio algorithm that identifies and predicts genes capable of encoding proteins in eukaryotic genomes, with a focus on fungi. It aims to extend the performance of its predecessor, GeneMark-ES, by integrating the process of aligning sequential RNA reads into the self-training process. It thus acquires greater accuracy in gene prediction.		LOMSADZE, Alexandre *et al*. Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. Nucleic Acids Research, [S.L.], v. 42, n. 15, p. 119-119, 2 jul. 2014. Oxford University Press (OUP). [22]
tRNAScan-SE	Software that identifies 99-100% of genes linked to RNA transport in DNA sequences, yielding less than one false positive per 15 Gb. Uses additional extensions to identify unusual carrier RNA homologues.	Perl	LOWE, Todd M.; EDDY, Sean R.. TRNAscan-SE: a program for improved detection of transfer rna genes in genomic sequence. Nucleic Acids Research, [S.L.], v. 25, n. 5, p. 955-964, 1 mar. 1997. Oxford University Press (OUP). [23]
InterproScan	Searches for proteins and amino acid sequences in a database called Interpro. Integrates predictive information about protein function, providing a broad view of the families to which the protein belongs and the domains and sites it contains.	Objective-C, Java	BLUM, Matthias *et al*. The InterPro protein families and domains database: 20 years on. Nucleic Acids Research, [S.L.], v. 49, n. 1, p. 344-354, 6 nov. 2020. Oxford University Press (OUP). [24]
EggNOG 5.0	Database covering the orthologous relationships and evolutionary history of genes, as well as annotations on features.	-	HUERTA-CEPAS, Jaime *et al*. EggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. Nucleic Acids Research, [S.L.], v. 47, n. 1, p. 309-314, 12 nov. 2018. Oxford University Press (OUP). [25]
Phobius	Prediction tool based on the hidden Markov model for transmembrane protein topology and peptide synthesis. It seeks to address the problems caused by the similarity between the helices of transmembrane proteins and sine peptides.	-	KÄLL, Lukas; KROGH, Anders; SONNHAMMER, Erik L.L. A Combined Transmembrane Topology and Signal Peptide Prediction Method. Journal Of Molecular Biology, [S.L.], v. 338, n. 5, p. 1027-1036, maio 2004. Elsevier BV. [26]
antiSMASH 5.0	Tool that identifies in the genome presented families of biosynthetic genes of secondary metabolites of bacteria and fungi. The new version has improvements in performance, maintenance and reliability of the results.	Python 3	BLIN, Kai *et al*. AntiSMASH 5.0: updates to the secondary metabolite genome mining pipeline. Nucleic Acids Research, [S.L.], v. 47, n. 1, p. 81-87, 29 abr. 2019. Oxford University Press (OUP). [27]
HMMscan	Part of a software suite called HMMER that compares a user-reported sequence (query) with the HMM database of the PFAM profile of protein families.	Perl, Python e Java	FINN, R. D.; CLEMENTS, J.; EDDY, S. R.. HMMER web server: interactive sequence similarity searching. Nucleic Acids Research, [S.L.], v. 39, n. , p. 29-37, 18 maio 2011. Oxford University Press (OUP). [28]
HMMsearch	Part of the HMMER software package responsible for matching an HMM profile against a sequencing database, covering several sequencing source formats.	Perl, Python e Java	FINN, R. D.; CLEMENTS, J.; EDDY, S. R.. HMMER web server: interactive sequence similarity searching. Nucleic Acids Research, [S.L.], v. 39, n. , p. 29-37, 18 maio 2011. Oxford University Press (OUP). [29]
SignalP	Artificial neural network that aims to recognise signal peptides and their cleavage sites, having a network for each of these tasks. It covers eukaryotes and prokaryotes.	-	NIELSEN, H. *et al*. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. Protein Engineering Design And Selection, [S.L.], v. 10, n. 1, p. 1-6, 1 jan. 1997. Oxford University Press (OUP). [30]
BlastP	Variation of the BLAST algorithm. Its function is to compare protein queries with databases to restore similarities between sequences.	-	ALTSCHUL, S. *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research, [S.L.], v. 25, n. 17, p. 3389-3402, 1 set. 1997. Oxford University Press (OUP). [31]

Hardware and operating system characteristics

For the development of the proposed configuration workflow, the hardware resources used were configured as follows:

- Processor: Intel® Xeon® CPU x5650 at 2.67 GHz x 12;
- Installed memory (RAM): 65.94 Gb;
- System type: 64-bit system, X64-based processor.
- Operating System: Ubuntu 20.04.2 LTS.

Among the operating systems, three stand out: MAC OS, Windows and Linux distributions, the latter being the most recommended for bioinformatics research purposes. Although less intuitive for users unaccustomed to open source systems, the variants present great flexibility and tools for practical purposes in the field. In addition, there are images and software developed for Linux that allow the automatic installation of programmes of interest or even help in the process, as is the case of Dugong and Unipro UGENE. There are many options for distribution, choosing Ubuntu in this article, because of the greater amount of information available on how it works and also because of the number of tools that support it.

Setting up the computing environment

The configuration of the computing environment involved the installation of the operating system, the configuration of the network environment, the firewall and the user account permissions. The chosen layout makes these steps clear and objective, without presenting major problems for the user. The greatest degree of difficulty is found in the use of the terminal, which is typical of Linux systems, but there is a wide range of manuals and guides on this in the virtual environment.

The next step must be the installation and configuration of the sequencing software, along with the necessary permissions to be able to operate fully. With these duly installed, it is possible to act on the management and processing of the memory dedicated to each one, finalising the configuration of the structure necessary to carry out the research related to the genome project. The software was chosen to cover the main steps of the sequencing process. FastQC [11], TrimGalore [12] and Kmergenie [13] were used for pre-processing. For assembly, SPAdes [5] and IDBA-UD [16] software were used to compare the difference between constant k-value and iterative k-value assemblies. Data evaluation was performed using Quast [17] and BUSCO [18] with reference genome. As for the annotation, only the initial part, referring to the structural annotation, was performed. The Augustus software [21], supported by the BLASTP tool [33], was used for this purpose. The datasets used are from the Staphylococcus aureus organism available on the SPAdes software

homepage [5], due to the existence of previous results, forming a partial validation mechanism for the process.

Results and Discussion

Initially, it is important to stress the great importance of creating a PATH variable to insert the programs to be installed and their dependencies, as well as the extremely important paths, such as the system's base folders. There are different ways to accomplish this task, such as using the "export" command in the terminal, which modifies the PATH variable only for the terminal in question, or modifying the file itself, in order to keep the changes. Regardless of the method chosen, there is a large source of detailed information on the process.

The installation of the mentioned software is similar. FastQC [11] and TrimGalore [12] are obtained from the same site. In both cases, the compiled .zip file is downloaded and extracted into the target directory. The program can be run via the terminal with the FastQC [11] folder directory defined or via PATH by the line ". /fastqc" or "fastqc", respectively. Attention should be paid to the Java version installed, because via PATH, other variants can be influenced so that the JRE has to be updated. It is best to install only the necessary dependencies according to a predefined list of programs to be used and their requirements. In the specific case of Java, you can run the command "java -version", obtaining the current version, and then "update-java-alternatives --list", which returns all the installed versions, inserted in the $PATH paths. Compared to the software version limiter, you can decide to keep it or change it using the line "sudo update-java-alternatives path/from/version".

TrimGalore [12] is executed by "trim_galore", either in the software's own path or via PATH, and its installation and execution did not present any other problems. In the case of Kmergenie [13] the download is via a "tar.gz" file. To perform the extraction, forward the folder to the destination directory and then enter the command "tar -xvf /directory", or similar. Then change the directory to the extracted folder and run the command make, which effectively installs the program, later executed via ". / kmergenie" or "kmergenie".

QUAST [17], despite having a browser version, can be installed, presenting in this format more variability and greater user control over the available parameters. Its installation is recommended, in order to enhance the learning of the tool and the understanding of the process as a whole. To work properly you need Python 2.5 and higher or Python 3.3 and higher, GCC 4.7 or higher, Perl 5.6.0 or higher, "GNU make" and "ar" and also the "zlib" package. It can be installed by the source file available on your site or by package managers, such as pip and linuxbrew. In the case of the source file, extract it into the target directory and then run the command line "quast.py" or "./quast.py" through the terminal. If it returns the software options, it is ready to use. To test the efficiency of the installation on a larger scale, run the command "quast.py -o /directory/to/results /quastdirectory/test_data/contigs_1.fasta". After checking the final message, which indicates the number of errors and warnings that appeared in the process. If no errors or warnings are returned, the program is ready for use. However, if it returns, you should check the "quast.log" file in the results directory to try to understand and resolve the problems. One warning that appeared several times during the development of this article was: "Cannot draw diagrams: python-matplotlib is missing or corrupted", referring to the lack of a specific python package, matplotlib, which is used in the creation of the graphics. After several tests of possible resolutions, a final and relatively simple conclusion was reached. The package in question was in a different version of Python run by QUAST [17]. To quickly solve the problem, the version of Python run by QUAST [17] presented in the initial lines of the terminal after executing the quast.py command was checked and found to belong to Python2. The command was then executed as "python3 quast.py", forcing it to run on Python3, the version on which the package was installed. This problem reinforces the importance of installing only the necessary dependencies when possible. You can try to work around this by assimilating the package to the Python2 version as well, but this would take more time and could cause obstacles in the future.

Another error encountered shows the output "'cgi' has no 'escape' attribute". To fix it, pay attention to the last line of the terminal before the error appears that interrupts the process and the directory it presents. Then find the file in the specified directory by opening it. Below the line "import cgi", add the line "import html", then find the line "cgi.escape", delete it and replace it with "html.escape". With these changes, QUAST [17] will run normally again. The latter problem seems to be more common when installing via pack manager pip. One interesting point worth mentioning is the fact that following this installation path for QUAST [17] also acquires the "--glimmer'' function, which already performs an initial prediction on the parsed sequence.

BUSCO [18] has several installation options, similar to those presented below for the annotation software and, as such, presents an extensive list of dependencies. It is executed by the busco command and has the configuration file call. This file contains information necessary for the proper functioning of the program, which must be edited before running it. It is recommended to specify all possible and relevant paths and parameters, thus ensuring a better performance of the program. An important point to note is the existence of optional and function-specific dependencies. In this case, it is up to the researcher to consider whether they are necessary or can be omitted in

this project.

When it comes to annotation-related software, a more dense field is entered. For the most part, they require a large number of installed dependencies, which may include other annotation software, as in the case of Maker. As a counterpoint, they often feature web versions, which have limitations, but provide a more intuitive mechanism and multiple modes of installation. The routes involving package managers, pip and anaconda, include the addition of the necessary dependencies, making the process quicker and easier. Another route available is through the docker images, mostly executed by the command "compile docker" in the terminal directory containing the so-called Docker file. It should be noted that the above options often have limitations linked to specific functions, which are outlined in detail in the read-only files accompanying the software. The final alternative is to acquire the software via source code. All paths have advantages and disadvantages, and it is up to the user to gauge their needs and capabilities and, based on these, choose the right direction. It is recommended that you carefully read the websites and read.me files of the programs to be installed, as they contain an installation guide for different operating systems and information necessary for the correct execution of their code.

A good starting point is with Augustus and HMMER, as these, as well as individual programs that present interesting and extremely important results for structural annotation, are dependencies of other more "complex" programs such as BUSCO. In some programs, such as Augustus, you can change files, in this case "common. mk", setting COMPGENEPRED, Add SQLITE and Add MYSQL to "false", to obtain a simple and fast version of the program, which does not lose performance, only some areas of performance that may be dispensable for certain genomes.

Due to the increase of dependencies and the diversity of applications, numerous version conflicts and errors can occur where already installed programs are missing because the directories are not included in the PATH variable. In programs with higher installation complexity, it is possible to perform a dependency check-list or even run the "make" command or the corresponding command that starts the assembly of the program, and wait for it to fail, because when this happens it issues an alert. .identify required files that are not specified. The executable files of the programs are installed in the /bin folder and, in case they are missing, you should consult the /src directory, always in the main folder of the software. So, in general, these are the paths to be added to the PATH variable. Below, you can see a table with the main commands of each chosen software, as well as some of general interest.

Table 2: Commands of interest for the present workflow.

Software	Command line	Function
FastQC	./fastqc ou fastqc	Executes the programme.
TrimGalore	trim_galore "opções" "caminho do arquivo"	Executes the program.
	trim_galore -h ou trim_galore --help	Displays the command list of the program.
Kmergenie	./kmergenie "caminho do arquivo"	Runs the program.
	./kmergenie	Displays extra information about the program.
SPAdes	spades.py	Displays the command list of the program.
	spades.py --test	Runs the dataset to test the installation.
	spades.py "opções" -o "diretório de saída"	Runs the program.
IDBA-UD	idba ou idba_ud	Displays information about the current version of the program.
Quast	fq2fa "arquivo.fq" "arquivo.fa"	Converts FASTQ readings to FASTA, from the bin/ fq2fa directory.
	./setup.py install ou ./ setup.py install_full	Installs the full basic version of the program.
	quast.py ou ./quast.py	Displays general information about the software.
	python3 quast.py	Forces the software to run through Python 3.
	quast.py -o "diretório de saída" "/diretóriodoquast/ test_data/contigs_1.fasta"	Tests the installation of the software.
BUSCO	busco	Displays general information about the software.
	busco --list-datasets	Displays the names of the datasets available for test execution.
	busco -i "arquivo" -l "nome/ caminho_do_dataset" -o "diretório_de_saída" -m "modo" "demais_opções"	Runs the BUSCO search.
General commands	export PATH=$PATH:/ caminho_a_ser_exportado	Adds certain directory to PATH for specific terminal.
	java -version	Displays the current version of Java.
	update-java-alternatives --list	Displays all installed versions of Java.
	sudo update-java-alternatives / diretório_da_versão_desejada	Toggles between previously installed Java versions.
	tar -xvf /arquivo_a_ser_extraído	Extract the tar.gz files in the directory where it is located.
	make	Execute the make file present in the directory, installing certain software.
	./build.sh	Mount the files of certain software.
	docker build .	Mount files from a dockable image present in the directory.

The part related to functional annotation will not be addressed in this article due to the high complexity present in this environment. To get started in this area, it is suggested to adapt to the use of BUSCO [18], due to the amount of dependencies and variable indexing system through the configuration file. In this way, the user will start to get used to the more computationally dense working environment and can then move on to more specific search and annotation programs. The use of the Maker programme can also be of great value to start the annotation process in general, as this software brings together several different steps in favour of the creation of genomic databases. However, its installation and execution is very complex. It works through four configuration files, as opposed to the single central file of Busco, and covers an extremely high number of dependencies, both mandatory and optional. In addition, it needs other programs, such as Apollo, to understand and study the results presented, programs that also have a more difficult installation flow.

Conclusions

The three general steps present very different degrees of complexity. Data pre-processing requires an understanding of the organism to be sequenced and how the experimental part of the process works. Basic level sequencing can be performed without much knowledge about the object of study, but optimal levels can be achieved with the addition of the same. In the case of annotation, it is essential to have extensive knowledge of the whole process logistics, both of the species and its phylogeny and of the programme to be implemented and the practical part of the study. This part of the project also often includes manual annotations, which requires special depth and focus.

At all levels of genetic sequencing, attention must be paid to the guides and text files provided by the programme developers, however, it is mainly the structural annotation programmes that require time to be set aside for reading. Through these you will discover the main lines of execution and the typical customisation options of the software, as well as its peculiarities. In addition, following or simply visiting the central Github page of the program, when available, can be extremely useful to identify possible bugs.

The comparison of assemblers, even if only between two programs, showed the essentiality of project planning, where software is chosen according to the needs presented. However, IDBA-UD [16] returned longer contigs, where the total length found for the genome was 2996254 base pairs, higher than that of SPAdes [5], which was 2977217 base pairs. In addition, the misassembled contigs presented by SPAdes [5] were more than five times longer than those presented by IDBA-UD [16]. Finally, the fraction of the genome constructed was 0.536 % higher through IDBA-UD [16], resulting in 98.799 %. However, it uses significantly more memory and time, and takes up to twice as long as SPAdes [5]. In view of the BUSCO analysis [18], the difference between the two assemblies was only 1 BUSCO, which appears fragmented in the assembly via SPAdes [5]. Therefore, between the two programs presented, SPAdes [5] is more economical and practical, while IDBA-UD [16] is denser and requires more powerful configurations, but is able to sequence with higher apparent accuracy.

On the structural annotation side, you could get interesting results with Augustus, although it runs at a basic level and without all the options that can be added to your pipeline. From the gene and protein sequences found, through BlastP [33], it was possible to generally evaluate the success of this portion. BlastP [33] identified different sequences presented by the programme as already studied sequences of the test organism, S. aureus.

The field of in silico gene sequencing is challenging for researchers who are not embedded in the environment. However, there is a growing number of tools that aim to facilitate the interaction of multiple user profiles from different projects. There is a great deal of flexibility and variability of programmes, allowing for an increasingly personalised choice appropriate to the resources available. By studying the theory and practice of the process and seeking to learn about the wide range of software available, satisfactory results can be achieved, which progress with the repetition and expansion of knowledge in the area.

References

1. **Borém, Aluízio; Santos, Fabrício Rodrigues.** *Biotecnologia Simplificada*. Viçosa: Editora Suprema, 2001.
2. **Sohn, Jang-Il; Nam, Jin-Wu.** *The present and future ofde novowhole-genome assembly*. Briefings In Bioinformatics, [S.L.], p. 1-18, 14 out. 2016. Oxford University Press (OUP).
3. **Angel, Victoria Dominguez del et al.** *Ten steps to get started in Genome Assembly and Annotation*. F1000Research, [S.L.], v. 7, p. 148, 5 fev. 2018. F1000 Research Ltd.
4. **Zerbino, Daniel R.** *Using the Velvet de novo Assembler for Short-Read Sequencing Technologies*. Current Protocols In Bioinformatics, [S.L.], v. 31, n. 1, p. 1-13, set. 2010. Wiley.
5. **Bankevich, Anton et al.** *SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing*. Journal Of Computational Biology, [S.L.], v. 19, n. 5, p. 455-477, maio 2012. Mary Ann Liebert Inc.
6. **Haas, Brian J et al.** *De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis*. Nature Protocols, [S.L.], v. 8, n. 8, p. 1494-1512, 11 jul. 2013. Springer Science and Business Media LLC.

7. **Stein, Lincoln**. *Genome annotation: from sequence to biology.* Nature Reviews Genetics, [S.L.], v. 2, n. 7, p. 493-503, jul. 2001. Springer Science and Business Media LLC.
8. **Zhou, Xiaofan et al.** *In Silico Whole Genome Sequencer and Analyzer (iWGS): a computational pipeline to guide the design and analysis of de novo genome sequencing studies*. G3 Genes|Genomes|Genetics, [S.L.], v. 6, n. 11, p. 3655-3662, 1 nov. 2016. Oxford University Press (OUP).
9. **Keith, Jonathan M**. *Bioinformatics*. [S.I]: Humana Press, 2017. 491 p.
10. **Ekblom, Robert; Wolf, Jochen B. W.** *A field guide to whole-genome sequencing, assembly and annotation*. Evolutionary Applications, [S.L.], v. 7, n. 9, p. 1026-1042, 24 jun. 2014. Wiley.
11. **Andrews, Simon et al.** *FastQC: a quality control tool for high throughput sequence data*. 2010.
12. **KRUEGER, Felix**. *Trim galore*. A wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files, v. 516, p. 517, 2015.
13. **Chikhi, R.; Medvedev, P.** *Informed and automated k-mer size selection for genome assembly*. Bioinformatics, [S.L.], v. 30, n. 1, p. 31-37, 3 jun. 2013. Oxford University Press (OUP).
14. **Luo, Ruibang et al**. *SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler*. Gigascience, [S.L.], v. 1, n. 1, p. 1-6, dez. 2012. Oxford University Press (OUP).
15. **Jackman, Shaun D. et al.** *ABySS 2.0: resource-efficient assembly of large genomes using a bloom filter*. Genome Research, [S.L.], v. 27, n. 5, p. 768-777, 23 fev. 2017. Cold Spring Harbor Laboratory.
16. **Peng, Y. et al.** *IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth*. Bioinformatics, [S.L.], v. 28, n. 11, p. 1420-1428, 11 abr. 2012. Oxford University Press (OUP).
17. **Mikheenko, Alla; Prjibelski, Andrey; Saveliev, Vladislav; Antipov, Dmitry; Gurevich, Alexey.** *Versatile genome assembly evaluation with QUAST-LG*. Bioinformatics, [S.L.], v. 34, n. 13, p. 142-150, 27 jun. 2018. Oxford University Press (OUP).
18. **Manni, Mosè et al.** *BUSCO update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes*. arXiv preprint arXiv:2106.11799, 2021.
19. **Bolger, Anthony M.; Lohse, Marc; Usadel, Bjoern**. *Trimmomatic: a flexible trimmer for illumina sequence data*. Bioinformatics, [S.L.], v. 30, n. 15, p. 2114-2120, 1 abr. 2014. Oxford University Press (OUP).
20. **Kim, Daehwan et al.** *Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype*. Nature Biotechnology, [S.L.], v. 37, n. 8, p. 907-915, ago. 2019. Springer Science and Business Media LLC.
21. **Keller, Oliver; Kollmar, Martin; Stanke, Mario; Waack, Stephan.** *A novel hybrid gene prediction method employing protein multiple sequence alignments*. Bioinformatics, [S.L.], v. 27, n. 6, p. 757-763, 6 jan. 2011. Oxford University Press (OUP).
22. **Lomsadze, Alexandre et al.** *Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm*. Nucleic Acids Research, [S.L.], v. 42, n. 15, p. 119-119, 2 jul. 2014. Oxford University Press (OUP).
23. **Lowe, Todd M.; Eddy, Sean R**. *TRNAscan-SE: a program for improved detection of transfer rna genes in genomic sequence*. Nucleic Acids Research, [S.L.], v. 25, n. 5, p. 955-964, 1 mar. 1997. Oxford University Press (OUP).
24. **Blum, Matthias et al**. *The InterPro protein families and domains database: 20 years on*. Nucleic Acids Research, [S.L.], v. 49, n. 1, p. 344-354, 6 nov. 2020. Oxford University Press (OUP).
25. **Huerta-Cepas, Jaime et al**. *EggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses*. Nucleic Acids Research, [S.L.], v. 47, n. 1, p. 309-314, 12 nov. 2018. Oxford University Press (OUP).
26. **Käll, Lukas; Krogh, Anders; Sonnhammer, Erik L.l.** *A Combined Transmembrane Topology and Signal Peptide Prediction Method*. Journal Of Molecular Biology, [S.L.], v. 338, n. 5, p. 1027-1036, maio 2004. Elsevier BV.
27. **Blin, Kai et al.** *AntiSMASH 5.0: updates to the secondary metabolite genome mining pipeline*. Nucleic Acids Research, [S.L.], v. 47, n. 1, p. 81-87, 29 abr. 2019. Oxford University Press (OUP).
28. **Finn, R. D.; Clements, J.; Eddy, S. R.** *HMMER web server: interactive sequence similarity searching*. Nucleic Acids Research, [S.L.], v. 39, n. , p. 29-37, 18 maio 2011. Oxford University Press (OUP).
29. **Finn, R. D.; Clements, J.; Eddy, S. R.** *HMMER web server: interactive sequence similarity searching*. Nucleic Acids Research, [S.L.], v. 39, n. , p. 29-37, 18 maio 2011. Oxford University Press (OUP).
30. **Nielsen, H. et al.** *Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites*. Protein Engineering Design And Selection, [S.L.], v. 10, n. 1, p. 1-6, 1 jan. 1997. Oxford University Press (OUP).
31. **Altschul, S. et al.** *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*. Nucleic Acids Research, [S.L.], v. 25, n. 17, p. 3389-3402, 1 set. 1997. Oxford University Press (OUP).
32. **Ucsd Jacobs School of Engineering.** *Single cell data sets*. Disponível em: http://bix.ucsd.edu/projects/singlecell/nbt_data.html. Acesso em: 01 jun. 2021.
33. **National Center for Biotechnology Information**. *Protein BLAST: search protein databases using a protein query. search protein databases using a protein query*. 2021. Disponível em: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Proteins. Acesso em: 20 maio 2021.