Simple RNA Sequencing Pipeline: A Nextflow Implementation

Tabea Attig mat.nr:6563671, Weronika Jaśkowiak mat.nr:6761084, Annika Maier mat.nr:6366281, Maike Nägele mat.nr:5474675

***Abstract*—The increasing availability of RNA-sequencing (RNA-seq) leads to a rising demand for efficient, reproducible, and scalable computational workflows. This project presents a modular RNA-seq analysis pipeline implemented in Nextflow, a powerful workflow management tool. The pipeline automates the process of RNA-seq data analysis, incorporating essential tools such as HISAT2, STAR, FastQC, Trim Galore!, Picard, and SAMtools. It efficiently handles raw sequencing data, performs quality control, read trimming, and alignment, while ensuring reproducibility through the use of containerization and version control.**

I. Introduction

Biomedical research has advanced over the years due to the rapid development of high-throughput technologies. Scientists around the world are conducting experiments and collecting complex data using established guidelines and well-developed pipelines. In science, it is crucial to ensure the reproducibility of experimental procedures and computational analysis.

In the field of computational biology, ensuring reproducibility is particularly challenging. To address this issue, workflow managers were developed. Additionally, the nf-core community is established to maintain developed workflows. Every nf-core pipeline is implemented in Nextflow, which allows for execution across diverse computational infrastructures and facilitates integration with container technologies such as Docker and Singularity [1]. The nf-core community develops a specialized framework that enables scientists to reliably reproduce pipelines across different research settings, such as universities and research institutions. This ensures that scientific workflows can be consistently replicated in any organization where scientific research is conducted, regardless of location or infrastructure [2].

Even minor changes in software versions, settings, or reference annotations can significantly affect results. Workflow managers address this issue by automating the tracking of input parameters and software tool versions within computational pipelines. This leads to improved consistency and reliability in research results. The nf-core community also encourages collaboration among researchers by providing a platform for sharing and improving pipelines. Collaborative work unites individuals from various backgrounds, bringing together a wide range of expertise. This joint effort speeds up innovation and ensures that the workflows are continuously updated. The nf-core community continues to expand and adapt each year to meet new challenges and innovations [1].

In particular, RNA sequencing (RNA-seq) has been and continues to be a vital technique in biomedical research [3]. It is an advanced sequencing method that allows for the comprehensive analysis of a biological sample's transcriptome and to uncover novel RNA species. The process starts with RNA extraction, which is then converted into complementary DNA and prepared for sequencing. This technique provides the quantification of gene expression levels, the discovery of novel transcripts, and the identification of alternative splicing patterns [4]. The RNA-seq is widely recognized as the gold standard for quantifying gene expression across the entire transcriptome, both in research and clinical settings [5].

Template credits: M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

II. Goal of the project

The small-scale project aims to develop a reproducible, simple RNA sequencing pipeline. This pipeline will include steps for assessing data quality, trimming raw reads, performing alignment, and finally marking duplicates.

III. Materials and Methods

To develop a simple RNA sequencing pipeline, six key steps will be considered: validating the user-provided sample sheet, assessing the quality of raw reads, removing adapters, and enhancing data quality, performing alignment, sorting the alignment output, and marking duplicates. As a reference, the existing RNA-seq pipeline is examined [6].

A. Nf-core pipeline general structure

The general structure of the nf-core pipeline can be broken down into four main components: `main.nf`, which executes the entire pipeline; `modules`, which contains specific processes for executing tools or scripts; `subworkflow`, which is a set of modules that work together toward a common purpose; `nextflow.config`, which includes parameters and different types of profiles, such as execution profiles [7].

B. Reference nf-core/rna-seq pipeline structure

The reference nf-core/rna-seq pipeline consists of five stages: pre-processing, genome alignment and quantification, pseudo-alignment and quantification, post-processing, and final quality control. In the first stage, the pipeline incorporates tools such as FastP [8], FastQC [9], Trim Galore!

[10], UMI-tools [11], and others. Specifically, the subworkflow `FASTQ_FASTQC_UMITOOLS_TRIMGALORE` handles quality control, UMI extraction, and trimming of RNA-seq reads. It uses several modules, including `FASTQC`, `UMITOOLS_EXTRACT`, and `TRIMGALORE`, to perform these tasks. The next steps involve alignment and quantification, followed by pseudo-alignment and quantification. The former includes tools such as STAR [12], HISAT2 [13], and SALMON [14], while the latter relies solely on SALMON for quantification. In the fourth stage, files are sorted and indexed using tools such as SAMtools [15]. The subworkflow `BAM_SORT_STATS_SAMTOOLS` is integrated to execute those tasks. Additionally, duplicate marking is performed using Picard [16]. The final stage generates comprehensive quality reports using tools such as DESeq2 [17] and Preseq [18] [6].

C. Testing data

For testing, the nf-core/rnaseq pipeline provides a test dataset available on GitHub [1]. The data originates from the yeast species *Saccharomyces cerevisiae*, consisting of 101bp paired-end, strand-specific RNA reads. This dataset was collected to investigate the mechanisms by which the transcription factor Rap1 regulates noncoding transcription and suppresses divergent transcription at gene promoters in yeast [19].

D. Development environment

The pipeline setup utilizes PyCharm v2023.3.+ [20], VisualStudioCode v1.86.+ [21], and the workflow manager Nextflow v23.10.+ [7].

E. Validation of sample sheet

A short Python script, `validation_samplesheet.py`, is implemented to validate the user-provided sample sheet. It is crucial to ensure that the input is in the correct format, preventing errors during execution.

F. Quality assessment

To assess the quality of raw reads, FastQC v0.12.1 is used by pulling the appropriate Docker container. This tool is a best practice for performing basic quality control checks on raw data [9]. It generates a final quality report for the sample in a file (.html), along with files (.zip) that contain the corresponding reports for the forward and reverse reads.

G. Adapter removal and enhancement of data quality

To eliminate adapters and enhance data quality, Trim Galore! v0.6.7 is employed by pulling the corresponding Docker container. Additionally, the same tool generates post-trimming quality assessments by incorporating the `--fastqc` parameter. It outputs trimmed single-end and paired-end files (.fq.gz), along with detailed files (.txt) and files (.html) containing the final quality reports. This step effectively cleans the data, guaranteeing that the reads are both relevant and of high quality.

H. Alignment

To compare sequenced reads to the reference genome and determine the most likely origin of each read, the STAR v2.7.10b and HISAT2 v2.2.1 are employed, by pulling respective Docking containers.

The HISAT2 uses a hierarchical indexing approach that combines the Burrows-Wheeler transform with the FM index to efficiently align RNA-seq reads. It constructs a global FM index for the entire genome, along with several local FM indexes that enable quick alignment extensions. HISAT2 handles reads in a single pass and at the same time gathers splice site information to align both long and short sequences with high sensitivity and speed. The advantage of the HISAT2 tool is that it requires only 4 GB of memory, making it ideal for use on standard desktop computers [13]. It produces a file (.sam), which includes details such as the position of each read on the genome, the quality of the alignment, and any mismatches or gaps.

The STAR aligner is an algorithm, written in C++. What makes it different from other aligners is its ability to directly align non-contiguous sequences, thereby detecting both canonical and non-canonical splice junctions, as well as chimeric transcripts. The algorithm consists of two parts: the seed-finding phase, which identifies a Maximal Mappable Prefix and the alignment phase, which constructs alignments for the entire read sequence by merging all seeds that were aligned to the genome during the initial phase. STAR is capable of fast mapping, processing up to 550 million paired-end reads per hour on modern multi-core servers. It combines speed with accuracy and scalability, making it suitable for analyzing large transcriptome datasets [12]. For small-size files, it can be used on standard desktop computers. STAR aligner produces an unsorted file (.bam) that contains information similar to the HISAT2 output.

I. Sorting alignment output

Before marking duplicates, sorting the file (.sam) is required. The tool SAMtools v1.20 is utilized for this step by using the corresponding Docker container. The sorting process arranges the alignment data in a specific order, usually by the position of reads on the reference genome. SAMtools outputs a sorted file (.sam).

J. Marking duplicates

Marking duplicates is essential for further downstream analysis, for instance for variant calling. Duplicates can arise during sample preparation, such as from PCR during library construction or from instrument failure. Especially in RNA-seq where duplicate sets are typically large, duplicate marking ensures that each RNA fragment is counted only once and removes biases from duplicate reads. For this step, the MarkDuplicates module of Picard v3.2.0-6 is used, by pulling the respective Docker container. The module compares sequences in a file (.bam) or (.sam) to identify and rank duplicates based on their base-quality. The tool generates a new file (.bam) that includes detailed information about each

read, indicating whether it is flagged as a duplicate, along with additional relevant data [16]. It also creates a metrics file detailing the number of duplicate reads.

IV. Results

A. Pipeline

The resulting pipeline can be seen in Fig. 1. It consists of six tools. In order to ensure modularity, each tool is called in its own module. The pipeline is intended to be run from the `main.nf` file. The pipeline creates a separate folder within the same directory, organizing the results of each tool into individual subfolders. The pipeline can be executed on any container specified using the profile flag, ensuring reproducible results regardless of the device used. However, running this pipeline within a Docker container is recommended, as it has been tested in that environment. Users can validate the pipeline by executing a test using the default test data provided prior to applying it to their own dataset. It is crucial to test both aligners by specifying the aligner option (`--align HISAT2/STAR`).

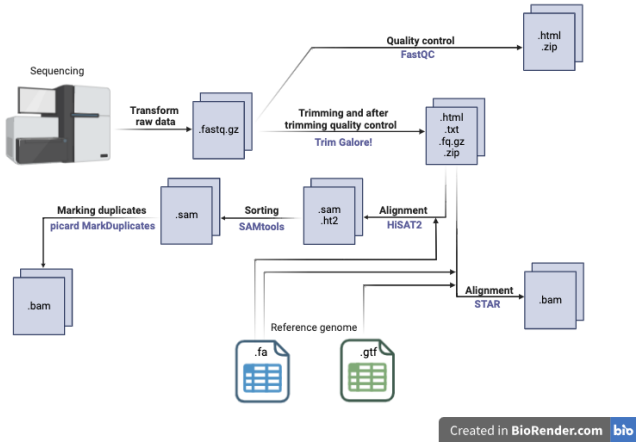

Fig. 1. Outline of the resulting pipeline

To start the test, the user can run the following command from the nf-core directory:

```
nextflow run main.nf
    -profile docker
    --align HISAT2/STAR
```

To run the pipeline for their analysis, the user must create a sample sheet including the correct paths to the data. Additionally, the user needs to specify the appropriate parameters for their analysis:

```
--samplesheet <PATH TO SAMPLESHEET>
--outdir <OUTDIR>
--fasta <GENOME FASTA>
--gtf <GTF>
--align <STAR>
--length <MINIMAL LENGTH OF TRIMMED READS>
--adapter <ADAPTER SEQUENCE>
```

The nf-core/ directory serves as the root folder of the pipeline, following the standard conventions of nf-core. Inside the root directory, the following components are present:

- `main.nf`: The main Nextflow script of the pipeline orchestrates the flow of data through each step, managing the entire process from raw sequencing reads to fully processed, aligned data. It specifies the order in which modules are executed, including tasks such as quality control, read trimming, and alignment with tools like HISAT2 or STAR. Default parameters are defined within the script, allowing users to customize the pipeline via the command line without needing to modify the underlying code,
- `nextflow.config`: The configuration file defines parameters, including maximal memory and maximal CPU usage. Additionally, profiles and registry settings for the Nextflow pipeline are defined. Modules reference profiles in the `nextflow.config` file, allowing them to automatically adjust parameters at runtime based on the selected profile,
- `bin/`: It includes an executable Python script, `validation_samplesheet.py`, used to validate the sample sheet. The sample sheet is passed to the `SAMPLESHEET_VALIDATION` process within the `samplesheet_validation` module via a channel. This process checks that the sample sheet for paired-end files (.fastq) meets specific criteria. It verifies the header format, ensures there are no missing values in any row, and confirms that both files (.fastq) are present and correctly named with the .fastq.gz extension. The function outputs a text file stating ”Samplesheet validation passed!” if successful, or ”Samplesheet validation failed!” with detailed error messages if validation issues are detected,
- `data/`: The pipeline’s test input data includes sample sheets in the `samplesheets/` directory, containing both valid (two or eight samples) and invalid sheets for validation testing. Additionally, the test data includes a reference genome for alignment, which includes a `genome.fa` containing the reference genome sequences and a `genome.gtf` that provides gene location and feature annotations. Paired-end sequencing reads in .fastq format are also provided for testing the pipeline’s alignment and downstream processes,
- `modules/`: Each module contains a `main.nf` file that outlines the processes to be executed in the workflow. These processes operate within a Conda environment specified in the `environment.yml` file, which outlines the necessary software and dependencies for the module.
  - `samplesheet_validation`: The validity of a sample sheet is checked using the `validate_samplesheet.py` script. The process runs this Python script, captures the output in a file named `validation.txt` and determines whether the validation passed or failed based on the contents of that file. If the validation fails, detailed error messages are both displayed in the console

and written to the `validation.txt` file. These messages specify the exact issues found for each sample (missing values, incorrectly named files,...),
  - `FASTQC/`: Runs FastQC on the paired-end files (.fastq) listed in the sample sheet to evaluate the quality of the sequencing data [9]. The process generates a FastQC report (.html) for each file (.fastq), along with a `versions.yml` file documenting the version of FastQC used in the analysis,
  - `trimgalore/`: This process performs read trimming on paired-end sequencing reads files (.fastq) using Trim Galore!, with optional adapter and minimum read length trimming based on user-specified parameters [10]. If provided, the `--adapter` option specifies the adapter sequence and `--length` sets the minimum read length for trimming. The process also runs FastQC for quality control, generating trimmed reads, reports, optional output files (.html), and compressed reports. Additionally, it generates a `versions.yml` file that logs the versions of the tools used during the process,
  - `hisat2/align/`: It consists of two processes: `HISAT2_BUILD`, which generates HISAT2 index files (.1.ht2 to .8.ht2) required for `HISAT2_ALIGN`, which align paired-end sequencing reads files (.fasta/.fastq) to the reference genome, producing an aligned file (.sam),
  - `STAR/`: This module consists of two processes: `INDEX_FILE` and `STAR_ALIGN`. The `INDEX_FILE` process generates a STAR genome index using a reference genome and file (.gtf), saving the index in a specified output directory to facilitate downstream alignment steps. The `STAR_ALIGN` process aligns paired-end reads to the indexed genome with STAR, producing sorted files (.bam). Additionally, it logs the versions of the tools used in a `versions.yml` file [12],
  - `picard_mark_duplicates/`: Executes duplicates step using Picard,
  - `samtools/`: Defines sorting step executed by SAMtools,
- `results/`: default directory to store output, i.e. `results/fastqc`.

The developed pipeline does not include any subworkflows, as seen in the reference nf-core/rna-seq pipeline in section III-B, due to its low complexity and the number of tools used.

B. Benchmarking

The benchmarking for the present RNA-sequencing pipeline was conducted on a machine equipped with 16 GB of RAM, an 8-CPU 64-bit system, running Ubuntu 22.04. For testing, two different sample sizes were employed: one paired-end sample and another set with eight paired-end reads. Modified, paired read samples from the pre-packaged dataset referenced in section III-C were utilized to create a subset of the original data. This adjustment effectively reduced the size and computational time needed for less powerful machines. `SRR63570707_1.fastq.gz` and `SRR6357070_2.fastq.gz` were used for the first sample. The larger sample was processed using the files (.fastq) from `SRR6357070` to `SRR6357077`.

The pipeline was run using the following command for the small sample set:

```
nextflow run main.nf
   -profile docker
   --outdir benchmark_test
   -with-report benchmark_results
```

The pipeline was run using the following command for the larger sample set:

```
nextflow run main.nf
   -profile docker
   --outdir benchmark_test_8_files
   --samplesheet 'data/samplesheets/
       samplesheet_8_samples.csv'
   -with-report
       benchmark_results_8_samples
```

The default path of the sample sheet and the reference genome, as well as the tool HISAT2 for alignment, were used. The total runtime for this process using the default files was **22.9 seconds**. During this run, processes such as `SAMPLESHEET_VALIDATION`, `FASTQC`, `TRIMMING`, and `HISAT2_BUILD` were able to execute in parallel since they do not depend on each other. However, as subsequent processes like `HISAT2_ALIGN`, `SAMTOOLS_SORT_AND_INDEX`, and `PICARD_MARK_DUPLICATES` depend on the output of the preceding processes, they are unable to take advantage of Nextflow's parallel processing capabilities. As illustrated in Figure 2, the TRIMMING step took the longest, which could have been a bottleneck in a traditional pipeline.

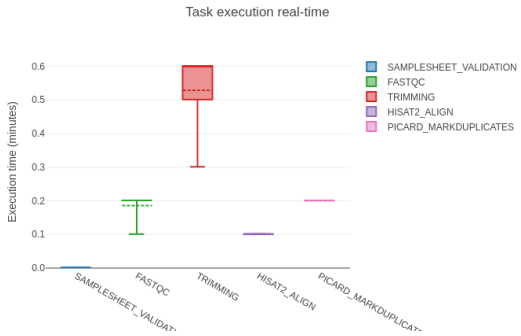

Fig. 2. Task execution time of the eight sample process

When running the pipeline manually with the same settings, it took approximately **14.8 seconds**, about ten seconds less than the Nextflow run. This difference can be attributed to Nextflow's overhead in setting up environments for each process. However, considering the time required to manually configure the environment for the manual run, Nextflow

would likely have been the faster option.

Additionally, a slightly larger dataset consisting of eight paired reads was utilized for further assessment. The sample sheet `samplesheet_8_samples.csv` was used along the standard reference genome, as well as the tool HISAT2 for alignment. The total runtime for this process with these settings was **59.7 seconds**. Similarly to the smaller run, the processes `SAMPLESHEET_VALIDATION`, `FASTQC`, `TRIMMING`, and `HISAT2_BUILD` were able to execute in parallel. The increased number of samples demonstrates the benefits of parallelization more clearly.

When executing the pipeline through the script `benchmarking.sh` with the same settings, including optional parameters to suppress console output, the process completed in **181.17 seconds** — approximately 80 seconds longer than the Nextflow run. Here the benefits of the parallelization become more evident and are expected to grow as the number of samples increases.

This analysis concludes that setting up a Nextflow pipeline may not be necessary for running just one or two files. However, it provides significant benefits when handling large datasets or repeatedly executing the pipeline on systems that can leverage Nextflow's multi-threading capabilities.

V. Discussion

A. Reproducibility and FAIR Pipelines with nf-core

The nf-core framework plays a crucial role in achieving reproducibility and compliance with FAIR (Findable, Accessible, Interoperable, Reusable) principles by incorporating containerization and strict version control. These features ensure that workflows can be executed in consistent environments across different platforms. The present RNA-seq pipeline incorporates aspects of accessibility and reusability. The Nextflow configuration file specifies different profiles, allowing users to set the package management system and container through the profile flag in the command line. Further, the `README` file in the GitHub repository ensures findability through a proper documentation. Results are organized modularly into subfolders, with each tool storing its outputs in a directory named after the tool.

However, several areas require further refinement to fully align with the FAIR principles. First, implementing tests to verify reproducibility, especially across diverse datasets, could improve the workflow's reliability. Moreover, introducing more robust early-stage error detection mechanisms for input files, would provide additional safeguards against data processing errors. Therefore, integrating metadata standards and generating standardized outputs are crucial steps.

B. Outlook

While the current pipeline is functional, well-documented, and user-friendly, several enhancements transform it into a more comprehensive and robust RNA-seq workflow. An immediate enhancement involves expanding the available parameters for each individual tool, thereby allowing for greater customization and flexibility. Furthermore, incorporating alternative tools for trimming and alignment in future iterations increase the versatility of the workflow, accommodating a broader range of use cases and research needs.

A critical area for future development lies in testing and validation. Implementing more rigorous error-handling mechanisms, particularly for file validation and edge-case scenarios, significantly enhance the pipeline's robustness. This ensures smooth execution across a wider variety of inputs, minimizing the likelihood of failures due to incomplete or corrupted data. Additionally, conducting more comprehensive tests on powerful infrastructure, such as high-performance servers are beneficial.

The modular design of the nf-core/rnaseq pipeline easily facilitates the integration of additional functionalities. Features such as pseudoalignment, transcript quantification, post-alignment, and more extensive post-processing steps can be seamlessly incorporated, leading to a more complex and capable RNA-seq data analysis workflow.

Current limitations, especially regarding local computational resources, can be mitigated by running the workflow on high-performance infrastructures like cloud computing platforms. This enables comprehensive benchmarking of the pipeline against the nf-core/rnaseq workflow, focusing on shared tools. However, these improvements are limited by available time and computational power.

VI. Conclusion

In this project, a small-scale modular and reproducible RNA-sequencing pipeline is developed using the Nextflow framework. The pipeline incorporates key tools such as HISAT2, STAR, FastQC, Trim Galore!, Picard, and SAMtools, ensuring efficient and automated data processing from raw sequencing reads to aligned data. By adhering to FAIR principles, the use of containerization and strict version control enhances reproducibility and portability. Benchmarking demonstrates the pipeline's efficiency, particularly in managing large datasets by taking advantage of process parallelization to improve runtime performance.

While the pipeline meets basic RNA-seq workflow needs, there are several areas for improvement to make it more robust and versatile. These include integrating more flexible parameterization, improving error handling, and expanding downstream analysis options, such as differential expression analysis. Implementing additional testing and validation procedures can further enhance reproducibility and ensure broader applicability across different datasets.

In conclusion, the implemented pipeline provides a strong foundation for RNA-seq analysis, offering efficiency, modularity, and reproducibility. Future enhancements can build on this framework to increase functionality, making the workflow more adaptable to a wide range of sequencing projects.

REFERENCES

[1] P. Ewels, A. Peltzer, S. Fillinger, H. Patel, J. Alneberg, A. Wilm, M. U. Garcia, P. Di Tommaso, and S. Nahnsen, "The nf-core framework for community-curated bioinformatics pipelines.," May 2022.

[2] W. A. . G. J. Wratten, L., "Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers," *Nature Methods*, vol. 18, no. 10, pp. 1161–1168, 2021.

[3] S. R. Ji F, "Rna-seq: Basic bioinformatics analysis," *Current Protocols in Molecular Biology*, vol. 124, no. 1, p. e68, 2018.

[4] M. S. Kukurba KR, "Rna sequencing and analysis," *Cold Spring Harbor Protocols*, vol. 2015, no. 11, pp. 951–969, 2015.

[5] L. M. M. J. e. a. Everaert, C., "Benchmarking of rna-sequencing analysis workflows using whole-transcriptome rt-qpcr expression data," *Scientific Reports*, vol. 7, no. 1, p. 1559, 2017.

[6] H. Patel, P. Ewels, J. Manning, M. U. Garcia, A. Peltzer, R. Hammarén, O. Botvinnik, A. Talbot, G. Sturm, nf-core bot, M. Zepper, D. Moreno, P. Vemuri, M. Binzer-Panchal, silviamorins, L. Pantano, R. Syme, G. Kelly, F. Hanssen, and C. Mertes, "nf-core/rnaseq: nf-core/rnaseq v3.16.0 - fire ferret (3.16.0)." https://doi.org/10.5281/zenodo.13882081, 2024. Zenodo.

[7] C. M. F. E. e. a. Di Tommaso, P., "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.

[8] S. Chen, Y. Zhou, Y. Chen, and J. Gu, "fastp: an ultra-fast all-in-one fastq preprocessor," *Bioinformatics*, vol. 34, p. i884–i890, July 2018.

[9] S. Andrews, F. Krueger, A. Segonds-Pichon, L. Biggins, C. Krueger, and S. Wingett, "FastQC." Babraham Institute, 2012.

[10] S. Andrews, F. Krueger, A. Segonds-Pichon, L. Biggins, C. Krueger, and S. Wingett, "Trim Galore!." Babraham Institute, 2012.

[11] T. Smith, A. Heger, and I. Sudbery, "Umi-tools: modeling sequencing errors in unique molecular identifiers to improve quantification accuracy," *Genome research*, vol. 27, p. 491—499, March 2017.

[12] F. S. J. D. C. Z. S. J. P. B. M. C. T. R. G. Alexander Dobin, Carrie A. Davis, "Star: ultrafast universal rna-seq aligner," *Bioinformatics*, vol. 29, no. 1, pp. 15–21, 2013.

[13] L. B. . S. S. Kim, D., "Hisat: a fast spliced aligner with low memory requirements," *Nature Methods*, vol. 12, no. 4, pp. 357–360, 2015.

[14] R. Patro, G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford, "Salmon provides fast and bias-aware quantification of transcript expression," *Nature Methods*, vol. 14, p. 417–419, Mar. 2017.

[15] P. Danecek, J. K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M. O. Pollard, A. Whitwham, T. Keane, S. A. McCarthy, R. M. Davies, and H. Li, "Twelve years of SAMtools and BCFtools," *GigaScience*, vol. 10, 02 2021. giab008.

[16] "Picard toolkit." https://broadinstitute.github.io/picard/, 2019.

[17] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for rna-seq data with deseq2," *Genome Biology*, vol. 15, Dec. 2014.

[18] T. Daley and A. D. Smith, "Predicting the molecular complexity of sequencing libraries," *Nature Methods*, vol. 10, p. 325–327, Feb. 2013.

[19] C. M. M. F. F. D. S. A. v. W. F. Wu ACK, Patel H, "Repression of divergent noncoding transcription by a sequence-specific transcription factor," *Mol Cell*, 12 2018.

[20] https://www.jetbrains.com/pycharm/.

[21] https://code.visualstudio.com/.