RNA-seq Pipeline

Jessie Midgley, MSc Bioinformatics
Jana Hoffmann, MSc Bioinformatics
Julia Graf, MSc Bioinformatics

October 23, 2024

1 Introduction

1.1 Pipelines

A pipeline is a structured sequence of processes designed to carry out data analyses in a systematic and repeatable manner. It typically comprises a series of interconnected steps, including data input, preprocessing, algorithmic computation, and result generation. The use of pipelines allows for the automation of repetitive tasks, ensures the reproducibility of scientific experiments, and optimizes the execution of processes to run efficiently on different types of computing systems, such as servers, clusters, or cloud platforms.

1.2 FAIR principles and reproducibility

The FAIR principles offer a guideline to improve the findability, accessibility, interoperability and reusability of scientific data and workflows [1]. Among other things, adding identifiers and extensive metadata to the data, as well as using standardized communication protocols, are part of the FAIR guiding principles. Meeting the requirements of the FAIR principles can assist to ensure reproducibility, which plays a major role in validating the results of scientific work.

1.3 nf-core

A major challenge in the bioinformatics community is achieving standardization, portability, and reproducibility of analysis pipelines [2]. Many pipelines are built as custom in-house scripts that are closely integrated with the computing environment in which they were developed. This can present challenges for others attempting to reproduce the results, which is a fundamental requirement for scientific findings. The

nf-core community project assembles a curated collection of best-practice analysis pipelines developed with Nextflow. Since nf-core pipelines are written in Nextflow, they can be run on various computational infrastructures, offer native support for container technologies like Docker and Singularity and allow for parallelization [3]. Pipelines within nf-core must adhere to strict guidelines, are required to provide high-quality documentation and must provide test dataset in order to run automated continuous-integration tests whenever there is a change to the pipeline [2]. A distinctive version tag is allocated with each pipeline release, thereby establishing a fixed link between the pipeline's implementation and the associated software dependencies. This way, nf-core assists in the development of high-quality pipelines that anyone can use.

1.4 RNA-seq

RNA-sequencing (RNA-seq) is a widely used technique for quantifying gene expression and allows for comprehensive profiling of the transcriptome [4]. One of the main goals of gene expression experiments is to identify transcripts that exhibit differential expression under various conditions. A usual RNA-seq workflow involves isolating RNA from the sample, converting it to complementary DNA (cDNA), and sequencing it using an NGS platform. The generated FastQ files contain sequencing reads and their associated base quality scores. These reads can then be aligned to a reference genome using mapping tools like STAR [5] or HISAT2 [6]. The alignment process allows for quantification of gene expression by counting the accumulation of aligned reads at specific loci in the reference genome [4]. The resulting read counts reflect the transcript abundance, which is used to determine gene expression levels [4]. Alternatively, alignment-free quantification methods, such as Salmon [7] or Kallisto [8], can be used to estimate transcript abundance without requiring a full read alignment.

2 Methods

2.1 Structure of nf-core pipelines

To ensure standardization, all nf-core pipelines have to be built using the nf-core template, which comprises many different files and folders. The most important files and directories are listed in Table 1 [9].

The workflow outlines the entire flow of the pipeline, calling different processes and subworkflows, and ensuring they are executed in the correct order. Subworkflows are smaller, reusable components within the main workflow. They should combine tools that work together as part of a specific subtask in the analysis [9]. A process represents a single, executable task in the pipeline. Each process usually runs a specific tool. These processes are defined in modules, which are independent, reusable

Table 1: Important files and directories of an nf-core pipeline.

Filename/Directory Name	Description
`main.nf`	The main Nextflow file which get executed when the pipeline is run. Calls workflows from the `workflows/` directory.
`nextflow.config`	The main Nextflow configuration file. It contains the default pipeline parameters, Nextflow configuration options and defines different configuration profiles that can be used to run the pipeline. Imports files from the `conf/` directory.
`README.md`	Contains basic information about the pipeline and usage.
`conf/`	Contains all of the configuration files.
`modules/`	Contains pipeline-specific and common nf-core modules.
`workflows/`	Contains the main pipeline workflows to be executed in the `main.nf` file.
`subworkflows/`	Contains smaller subworkflows that typically consist out of a few modules chained together.

Nextflow scripts. Modules allow for standardization and can be reused across different pipelines. Channels pass data between these modules, ensuring that the outputs of one process connect to the inputs of another.

2.2 Tools and computing environment

The pipeline is implemented in Nextflow and utilises the tools listed in Table 2. The

Table 2: Tools used in the pipeline and their version.

Tool	Version
FastQC	0.12.1
picard MarkDuplicates	3.2.0-1-g3948afb6b
RSEM	1.3.1
Salmon	1.10.3
STAR	2.7.11b
Trim Galore!	0.6.10
Cutadapt	4.9
RNAseq	v1.1.0
Nextflow	24.04.4

pipeline automatically installs all required tools with Docker containers, ensuring that each tool runs in a consistent computing environment across all devices [10]. Each process is executed within its own Docker container, and the Docker images

used to build these containers are sourced from Seqera [11]. To maintain compatibility across both ARM and AMD architectures, users can specify their processor type, allowing the retrieval of the appropriate Docker images. For testing, the pipeline was executed locally on our laptops.

3 Results

Our pipeline follows the essential steps of reading in a samplesheet, performing quality control, trimming and aligning reads to the reference genome, quantifying transcript abundance, and marking duplicate reads. The samplesheet refers to a CSV file in which each row represents either a single-end FastQ file or two paired-end FastQ files, with a corresponding sample identifier. Quality control is then performed on the sequencing files listed in the samplesheet, using the tool FastQC [12]. FastQC conducts a series of analyses on the raw sequencing data to give an overview of potential issues, presenting summary graphs and tables for easy visualisation. Following quality control, the reads are trimmed with Trim Galore! [13] to remove adapters and low-quality bases from the ends of the sequencing reads, resulting in compressed FastQ files. The trimmed sequences are then aligned to the reference genome, producing compressed binary alignment map (BAM) files. This process is executed within a subworkflow that first utilizes STAR [5] to index the reference genome and subsequently aligns the reads to it. This subworkflow is divided into two processes, which allows for parallelization of the read alignment step. Following alignment, the transcript quantification is performed in a second subworkflow consisting of three processes. First, reference transcripts are extracted from the genome by RSEM [14], which prepares the reference by using a provided genome annotation file in gene transfer format (GTF). In a second process, Salmon [7] is used to index the reference transcripts generated by RSEM. Finally, the indexed reference transcripts, along with the sample reads, are processed by Salmon to produce transcript abundance tables in the third step of the alignment subworkflow. For each input sample, a TSV file is generated containing the transcript name, its length, its abundance in terms of Transcripts Per Million (TPM), and the estimated number of reads originating from each transcript. The outputs of each process can be found in the `results/` directory, under the corresponding subdirectory of the tool used by the process. Additionally, each module in the workflow emits a `versions.yml` file, containing the version number for each tool executed by the module. A summary of all software versions used by the pipeline can be found in `results/pipeline_info/pipeline_software_versions.yml`.

4 Discussion

4.1 Reproducibility and FAIR principles

For every nf-core pipeline, it must be possible to run the pipeline with `--profile docker`, ensuring that all software dependencies are fully met [2]. Additionally, every container has to be versioned with a specific, stable version. This ensures that reproducing an analysis in a different computing environment is straightforward, requiring only the use of the correct stable release tag. To find the matching nf-core pipeline for a specific use case, users can search by keywords on the nf-core website. All the nf-core pipelines are open source and released under the MIT license, making them freely accessible. A centralized repository of well-documented pipelines allows reusing pipelines rather than rewriting them.

Our pipeline also uses containerization with Docker, and all the containers have a stable version tag. The pipelines also creates a `pipeline_software_versions.yml` file which contains the versions of all software used in the pipeline. Our pipeline is compatible with both ARM and AMD architectures. When run with the `--arm` option, it leverages a different set of Docker containers than when executed with the `--amd` option. As a result, cross-platform reproducibility cannot be fully guaranteed due to variations in the underlying computing environments. However, within the same architecture, reproducibility is ensured by using identical Docker containers to maintain a consistent execution environment. The different containers across platforms install the same version of the employed tools to minimize platform-specific differences. The source code for our pipeline is publicly available via GitHub and released under the MIT license which makes it accessible for everyone. Our documentation enables users that are not familiar with the pipeline to get started on using it.

4.2 Outlook

While the pipeline successfully generates transcript quantification data, there are opportunities to improve its performance by incorporating additional processes. For instance, integrating support for Unique Molecular Identifiers (UMIs) would allow for their extraction and downstream read deduplication, improving the accuracy of the results and reducing potential biases. The strandedness of sequencing reads could be inferred automatically by combining the subsampling of FastQ files with pseudoalignment, to determine whether reads correspond to the original RNA sequence or its complementary cDNA. Stranded RNA-Seq data offers advantages over non-stranded data, by improving the accuracy of transcript assembly and differential expression [15]. Additionally, the removal of rRNA from reads can lead to the better detection of mRNA transcripts, which is important for the analysis of differential expression [16]. Finally, the generation of additional bigWig coverage files would allow for clearer visualisation of transcript coverage across the genome.

Data Availability

The source code for this paper is available on GitHub.

References

[1] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[2] P. A. 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," *Nature biotechnology*, vol. 38, no. 3, pp. 276–278, 2020.

[3] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.

[4] K. R. Kukurba and S. B. Montgomery, "Rna sequencing and analysis," *Cold Spring Harbor Protocols*, vol. 2015, pp. 951–969, April 13 2015.

[5] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras, "Star: ultrafast universal rna-seq aligner," *Bioinformatics*, vol. 29, p. 15–21, Oct. 2012.

[6] D. Kim, J. M. Paggi, C. Park, C. Bennett, and S. L. Salzberg, "Graph-based genome alignment and genotyping with hisat2 and hisat-genotype," *Nature Biotechnology*, vol. 37, p. 907–915, Aug. 2019.

[7] 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.

[8] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, "Near-optimal probabilistic rna-seq quantification," *Nature Biotechnology*, vol. 34, p. 525–527, Apr. 2016.

[9] nf-core contributors, "Pipeline file structure." `https://nf-co.re/docs/contributing/pipelines/pipeline_file_structure`, 2024. Accessed: 2024-10-11.

[10] "Use containers to build, share and run your applications." `https://www.docker.com/resources/what-container/`. Accessed: 2024-10-17.

[11] "Containers." `https://seqera.io/containers/`. Accessed: 2024-10-17.

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

[13] F. Krueger, F. James, P. Ewels, E. Afyounian, M. Weinstein, B. Schuster-Boeckler, G. Hulselmans, and Sclamons, "Felixkrueger/trimgalore: v0.6.10 - add default decompression path," 2023.

[14] B. Li and C. N. Dewey, "Rsem: accurate transcript quantification from rna-seq data with or without a reference genome," *BMC Bioinformatics*, vol. 12, Aug. 2011.

[15] B. Signal and T. Kahlke, "how_are_we_stranded_here: quick determination of rna-seq strandedness," *BMC Bioinformatics*, vol. 23, Jan. 2022.

[16] M. M. Pastor, S. Sakrikar, D. N. Rodriguez, and A. K. Schmid, "Comparative analysis of rrna removal methods for rna-seq differential expression in halophilic archaea," *Biomolecules*, vol. 12, p. 682, May 2022.