Small-RNA sequencing analysis pipeline in Snakemake and comparison to Nextflow

Florian Radziej

A thesis presented for the degree of
Bachelor of Science

Algorithmic Bioinformatics
Heinrich Heine University Düsseldorf
Germany
4th August, 2022

Acknowledgments

I would like to express my gratitude towards Prof. Gunnar Klau for giving me the opportunity to write this thesis. Many thanks to Prof. Dr. Tobias Marschall for being the second assessor. Moreover a special thanks to my daily supervisor Sarah Schweier for her help in weekly meetings, advice on questions and for giving me feedback during the bachelor thesis and especially on my writing. I would also like to thank Johannes Köster and the nf-core community, who helped me with questions about the pipelines.

Abstract

Workflow management systems represent, manage, and execute multi-step computational analyses and offer many benefits to bioinformaticians. In this thesis, we compare Snakemake and Nextflow on the basis of the completed pipeline SmRNASeq. The given pipeline is a best-practice Small-RNA sequencing pipeline based on Nextflow and was translated as part of this thesis into Snakemake. For this purpose, we examine the different approaches of the workflows, point out similarities and differences and compare the runtime.

Contents

1 Introduction

Small-RNA sequencing analysis, especially the analysis of microRNA, has gained significant interest in recent years. microRNAs play a central role in the regulation of gene expression [1]. The Institute of Pathology of the UKD [2] wants to analyze microRNAs as potential biomarkers to distinguish between Intraductal Pancreatic Mucinous Neoplasms, a certain type of pancreas lesion, with different degrees of dysplasia, that describes the severity of the lesion. However, transforming biological data into information involves running a large number of tools. Individual execution of tools is time-consuming and difficult to reproduce. Therefore, it makes sense to build automated pipelines to get reproducible and scalable results. An available pipeline for this purpose is SmRNASeq [3], a small-RNA sequencing analysis pipeline.
The idea of a pipeline is to chain different tools together to optimize resource usage, be able to run across multiple platforms, simplify sharing and get highly reproducible results. Although different pipeline tools serve a similar purpose and the analysis process should have the same end result, the choice of pipeline tools may offer different advantages and application options. This makes the comparison between workflow management systems worthwhile and also motivates the topic of this bachelor thesis.
First, we present the biological and statistical foundation that is needed in this thesis. We use the pipeline SmRNASeq [3] based on the Nextflow [4] framework for the comparison. To do to so, we implemented it in Snakemake [5] for the purpose of this thesis. Furthermore, we examine the different approaches of the pipelines, show similarities and differences between them and compare the runtime.
We concluded that both pipelines are adequate tools to create pipelines, but they differ in terms of features and detailedness. Nextflow offers more options to design a workflow, the underlying concept was more intuitive for us, and is easier to extend for more computationally intensive pipelines.

2 Preliminaries

The following section provides the necessary background knowledge to understand the function of the workflows of this thesis.

2.1 MicroRNA

MicroRNA (miRNA) are small, single-stranded RNA molecules that have been shown to play a central role in post-transcriptional regulation of gene expression and suppression of gene transcription. Their regulatory role has been associated with multiple types of cancer and several other complex diseases such as diabetes. MiRNAs are a promising candidate for biomarker development and therefore of particular interest to the research of cancer diagnosis. [1] [6]
The canonical miRNA biogenesis pathway is shown in Figure 1 [1]. MiRNA host genes are transcribed by RNA polymerase II in the cell, which yields the primary miRNA (pri-miRNA). The pri-miRNA requires two endonuclease processing steps before it becomes a mature miRNA. First, the pri-miRNA binds to a protein complex named Drosha which forms a hairpin loop structure of the primary miRNA. The resulting short hairpin RNA is called precursor-microRNA (pre-miRNA), hereafter referred to as "hairpin". The precursor miRNAs are released into the cytoplasm by the nuclear export protein Exportin 5. In the cytoplasm, pre-miRNA undergo cleavage by the enzyme Dicer into about 22 nucleotide-long mature miRNAs. [7]
The miRNA sequencing data, which is analyzed in the workflows, contains sequences of mature and hairpin miRNA.

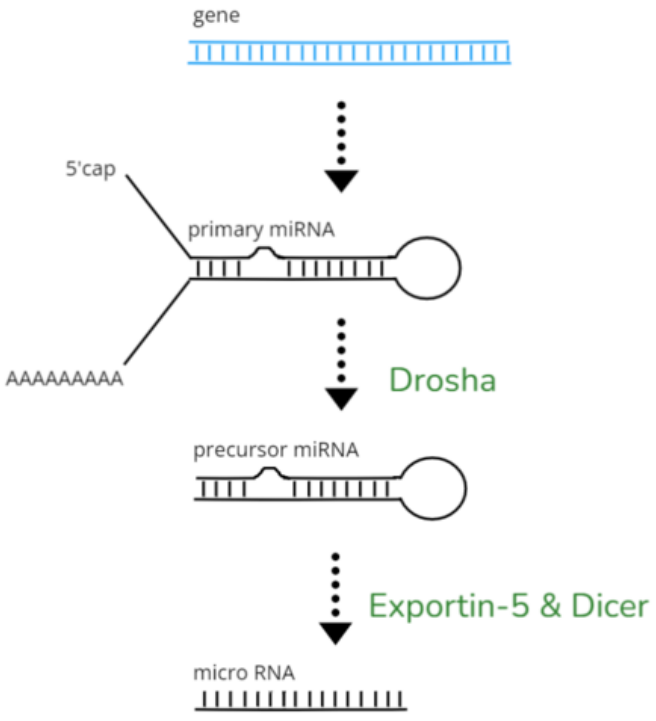

Figure 1: micro RNA biogenesis

2.2 Intraductal Pancreatic Mucinous Neoplasms

Pancreatic ductal adenocarcinoma (PDAC), a type of pancreas cancer, is one of the most deadly diseases with little success from treatment therapies. PDAC has a survival chance of 9 percent within a period of 5 years [8]. Intraductal Pancreatic Mucinous Neoplasms (IPMN) are cystic neoplasms of the pancreas. They can be classified as precursor lesions of PDAC and are shown in Figure 2. [6]

IPMNs can be divided into histological subtypes according to their expression level of mucins, a family of proteins. The subtypes are called in order of highest to lowest prevalence: gastric, intestinal, and pancreaticobiliary. Beyond that, pancreatic precursor lesions are classified based on their degree of dysplasia, which divides the lesions low- and high-grade categories. [6] Individual therapies have limited success because high-risk cases are not detected and unnecessary surgery on low-risk IPMNs is performed. Therefore, the development of biomarkers that predict malignant behavior is desired. In general, biomarkers are measurements of structures and processes that help to diagnose diseases. [9] In search for potential biomarkers, the Institute of Pathology of the UKD [2] has come across miRNAs, which are considered in this thesis.

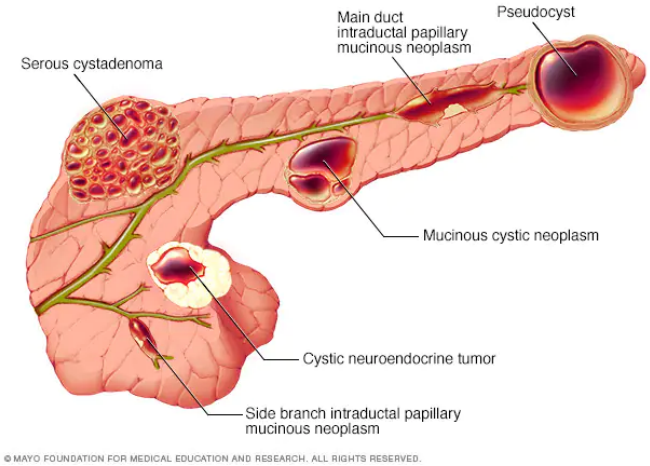

Figure 2: Pancreas with two Intraductal Pancreatic Mucinous Neoplasms lesions [10]

2.3 RNA-Sequencing

RNA sequencing (RNA-seq) is a procedure that combines the quantification of gene expression and transcript identification. The utility of RNA sequencing is widely recognized in the genomics community and has become a standard toolkit of the research community. Small-RNA sequencing includes three major steps: isolation of the RNA, preparation and construction of the cDNA library, and finally sequencing as shown in Figure 3. [11] First, small-RNA is isolated from biological samples by conventional RNA extraction methods. To guarantee success, the RNA should be of sufficient quality which can be ensured by filtering the length separation or binding of small RNAs to specific proteins. The preparation of the cDNA library consists of extending the miRNA by ligation, which introduces primer-binding sites because

of the short length of the miRNA, followed by reverse transcription to obtain complementary DNA (cDNA). The resulting cDNA is amplified by PCR (Polymerase Chain Reaction) to receive a cDNA library. The library is now ready and sequencing is performed on a high-throughput platform, commonly Illumina [12]. Sequencing data is stored in FASTQ [13] files. Since FastQ files consist of millions of entries, they are compressed due to their size. An entry consists of an identifier with information about the sequence, the sequence itself, and quality scores.

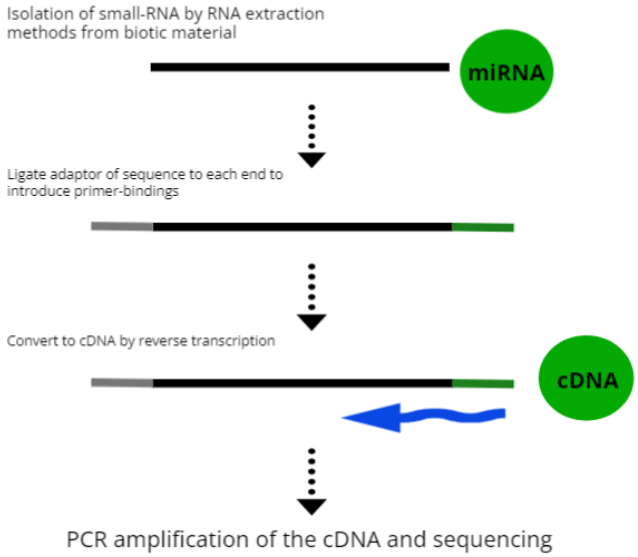

Figure 3: Small RNA sequencing procedure

RNA sequencing can be integrated into an analysis process. Often used steps for a typical RNA-seq analysis are quality control of raw reads by analyzing the sequence quality, alignment of the sequencing reads to a reference genome, filtering and normalizing between samples, and statistical modeling of significant changes in the expression of genes between sample groups, such as differential expression analysis. [14]

2.4 Differential Expression Analysis

Differential expression analysis is a type of statistical analysis that allows finding differences in gene expression between two or more types of samples. As an example, one could compare healthy tissue and tissue of a patient with a disease. In this thesis, the samples are different types of pancreas tissue provided by the Institute of Pathology of the UKD [2]. The analysis is based on normalized read counts. The goal of the differential expression analysis is to consider the magnitude of expression between groups of samples and identify the significance of the differences in the end result. For each miRNA, it is decided whether it is differentially

expressed between the groups of samples. For this type of analysis, tools such as edgeR [15] or DESeq [16] are available. Both tools are implemented in the programming language R [17], making extensive use of statistical analysis capabilities the language itself and its respective ecosystem of libraries provides.

3 Pipeline Implementation

This section introduces the pipeline tools for data analysis, Snakemake and Nextflow. Additionally we describe the small-RNA sequencing analysis pipeline SmRNASeq, and its Snakemake implementation, which was done as part of this thesis.

3.1 Workflow Management Systems

Workflow management systems (WMS), such as Snakemake and Nextflow, have been developed to manage computational data-analysis workflows. WMS enable a user to chain programs that depend on each other without having to call them individually for every execution. The goal of WMS is to simplify the development and execution of processes, remove the complexities of sharing a given pipeline with colleagues, and monitor resources easily.
The main characteristics of workflow management systems include *execution management*. Each workflow task is managed by the operating system and processes are split to run in parallel to shorten the runtime. To ensure high *portability*, workflows can be run on different computing infrastructures by tweaking simple parameters. Software and their dependencies are managed by *environment management systems* like Conda [18] or Charliecloud [19], such that the workflow runs reliably in different computing environments. The use of environment management systems and explicit parameter specification ensure high *reproducibility*, which means the workflow will produce the same result when running on different computing platforms. [4] [5]

3.2 Tools Used to Setup and Build the Pipelines

3.2.1 Conda

Conda [18] is an open-source package and environment management system. It allows to create self-contained environments. A user can easily save, load and switch between them. Conda, and specifically Bioconda [20], are used for version control of the utilized programs, leading to a simple installation. Conda analyzes each package for compatible dependencies, and how to install them without conflict. If there is a conflict, Conda will inform that the installation cannot be completed. Encapsulating the individual rules and environments allows the user to install and update various packages and their dependencies, which supports simple sharing of rules between workflows and easy adaption and extension of existing workflows. Furthermore, it avoids having compatibility issues between dependencies that are not needed in the same rule. [18]

3.2.2 Trim Galore

Trim Galore [21] is a package that uses Cutadapt [22] for adapter trimming and quality trimming. Once trimming has completed, FastQC [23] can be run on the resulting output for quality control.

3.2.3 Bowtie

The bowtie [24] package allows to align large sets of sequencing reads to a reference sequence in a fast and memory-efficient way. The package includes tools to build indexes for references that are used for the alignment.

3.2.4 Samtools

SAMtools [25] is a library and software package for parsing and manipulating alignments in the SAM format. The SAM (Sequence Alignment Map) format is a generic format for storing large nucleotide sequence alignments. The tool provides various utilities including sorting, merging, indexing and generating alignments in a per-position format.

3.2.5 FastX Toolkit

The FASTX-Toolkit [26] consits of a collection of tools for pre-processing data from FAST/Q files.

3.2.6 Seqkit

Seqkit [27] is a toolkit for FASTA/Q file manipulation. It supports plain or gzip-compressed inputs and outputs from either standard streams or local files.

3.2.7 EdgeR

EdgeR [15] is an R [17] package based on the open source project Bioconductor [20]. It implements a range of statistical techniques including empirical Bayes estimation, exact tests, generalized linear models and quasilikelihood tests for differential expression analysis of RNA-sequencing expression data.

3.3 Nextflow

Nextflow [4] is a workflow management system that is built to implement portable, scalable, and reproducible workflows. It chains the execution of multiple command-line applications and tools.
Nextflow uses domain-specific language (DSL) extensions of Groovy to simplify writing workflows. Groovy [28] is a dynamic language that runs on the Java [29] platform. This increases the flexibility of Nextflow, as steps in a pipeline can be expressed via DSL. Cases that are difficult to implement can be solved using Groovy. A step in a workflow is defined as a *process*. Commonly, each process contains an input, output, and script block. At minimum, a process must contain a script block that defines the command to be executed. A script block can be written in any Linux-executable scripting language such as Bash, Python, or Ruby. Processes are isolated from each other but can communicate by sending values and files from input to

output block via channels. Channels are asynchronous FIFO queues, which means channels do not produce outputs in the same order as the associated inputs and elements in the channel are processed according to the FIFO principle. This allows processes to run as soon as they receive input from a channel. Nextflow allows modifying elements in a channel by the use of operators, for example to select certain elements by a filter. Parallelization in the workflow execution is an implicit consequence of the way inputs and outputs from each process are channeled to other processes. Nextflow defines complex interactions and accessible parallel computation environments based on dataflow programming model. The data flow model interprets processes as a series of connections until a final output is reached.
The configuration of the pipeline can be decoupled from the workflow implementation to ensure a portable workflow that can run on any computational environment. It is considered best practice to store the settings in the file nextflow.config. In the configuration file parameters like CPU load and memory usage or specification of process parameters can be defined and adjusted. Processes can be labeled allowing the same configuration to be applied to all groups of processes. Nextflow has by default a working directory for each task within a parent directory of the run. Unique folder naming is ensured by using long hexadecimals, names of the workflow stages, or execution timestamps. The exact structure and names of the folders vary and the user has no control over these parameters. But with the *publishDir* annotation in a process, selected results can be saved to a specified folder. Nextflow's workflow abstraction separate between workflow definition and underlying execution environment, that can be achieved with the *executor*. The executor determines how and with what properties the pipeline runs on the specified system. To manage the version of software used in the workflow, it is possible to choose between Conda [18], Docker [30], Singularity [31], Charliecloud [19], Podman [32], and Shifter [33] with little effort in the configuration file, parameters can be tweaked to execute the workflow in different environments such as HPC clusters, cloud computing resources as offered by Amazon Web Services [34] and similar providers, or locally. [4]

3.3.1 SmRNASeq

The SmRNASeq [3] pipeline is a project from nf-core [35]. Nf-core is a bioinformatics community that collects and composes curated sets of best-practice analysis pipelines created with Nextflow. All pipelines from nf-core have the goal to ensure portability, reproducibility, and conflict free execution.
The Nextflow pipeline SmRNASeq from P. Ewels et al. explained in this section is a best-practice small RNA-sequencing analysis pipeline. Figure 4 shows the structure of SmRNASeq in form of a directed acyclic graph, an generated output of the pipeline. Each node represents a process and each edge is a channel, through which the connected processes exchange information. Smaller nodes represent operations on the channels, while the arrows show the direction of the data flow. The pathway highlighted in green includes preprocessing of the input data,

alignment steps and statistical analysis of the aligned reads. It is implemented as part of this thesis in Snakemake. The processes on the right side of Figure 4 are responsible for further quality control steps specific to small RNA sequencing and creating a summary of the results in an HTML report. In addition, the mirdeep process is used for the identification of new and known miRNAs in deep sequencing data [36]. We now describe the green path in more detail. First, the workflow reprocesses the reference data for the hairpin and the mature miRNAs. The *bowtie_indices* process prepares the reference data for the next step by unzipping the data if necessary and replacing special characters. After that, hairpin and mature miRNAs are selected for the given species, the individual RNAs are transformed to DNA and a bowtie index is built, which consists of a constant number of six files. In the middle of the Figure 4, the process *trim_galore* receives the input data, in our case, files in fastq.gz format containing miRNA sequencing read data. The tool Trim Galore! uses Cutadapt to remove adapter contamination. After completion, the quality of the reads is checked by FastQC. Furthermore the trimmed reads are passed to the next process *bowtie_miRBase_mature* which uses Bowtie to align the newly build reference bowtie index of the mature miRNA to the trimmed reads. Reads, which could not be aligned to the given reference are forwarded to *bowtie_miRBase_hairpin*. Here, the remaining reads are aligned to the hairpin reference bowtie index. The tool Samtools provides various utilities to index, sort, and generate comprehensive statistics from the alignment data. In the process *mirna_post_alignment* it is used to prepare the aligned data from both alignment processes for the analysis step. The counts of miRNAs found for each sample are then passed to *edge_miRNA*. This process calls an R script and hands over a hairpin and a mature stats file for each sample from the previous process. Expression analysis is performed by the script, which produces statistics in the form of graphs and tables.

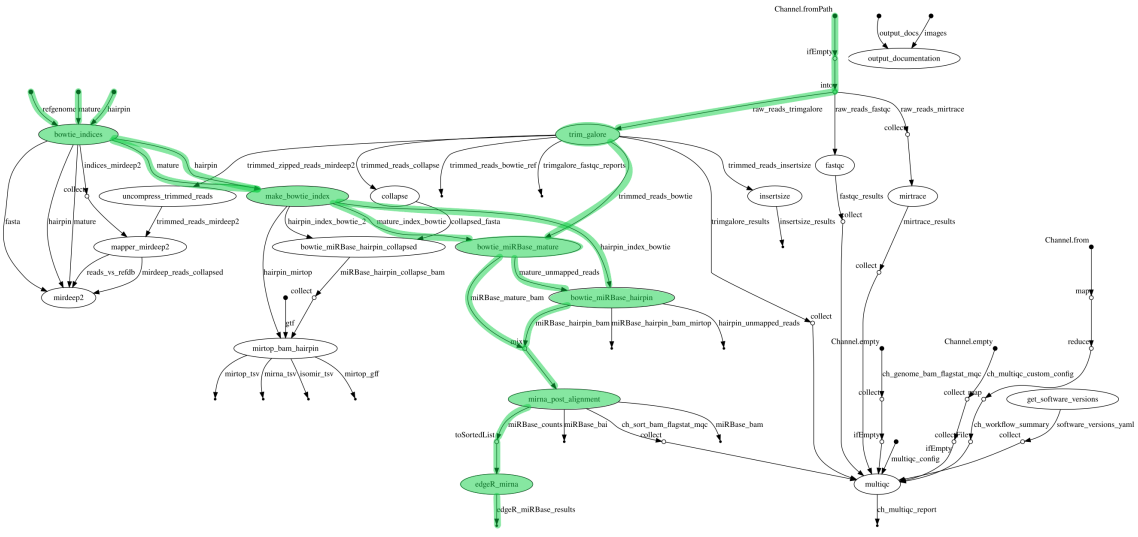

Figure 4: Directed Acyclic Graph of SmRNASeq

3.4 Snakemake

Snakemake [5] is a tool for workflow management to create reproducible and scalable data analysis. In simplified terms, a Snakemake pipeline is a chained execution of different command-line applications and programs. Its domain specific language (DSL) is implemented as an extension of the programming language Python [37], which adds the benefit of readability. Advantages of the underlying language are still available. The individual workflow steps are built via so-called rules. Each rule consists of a name and a set of directives. Although not strictly required, most rules contain an input and output directive, which specify filenames. For the instruction of a rule multiple directives can be used. The script-directive is used to run external scripts or regular Python code, being able to access directly using all properties of the job. To execute Python code as well as shell commands one can use the run-directive, while the shell-directive is restricted to executing shell commands only. Input and output files can contain so-called wildcards, which define the file names by generalizing parts of the names. This reduces repetition and increases the readability of the code. Rule dependencies are implicit, since for each file used by a job, Snakemake looks for a rule that generates it by substituting all existing wildcards and mapping input files to output files. Snakemake continues recursively until all input files of all jobs have been created by another job unless they already existed in the used storage. When Snakemake is invoked without a specific target, the first rule (the input-only rule *all*) is executed. The input files of the rule *all* are usually the final outputs from the pipeline. Snakemake outputs a directed acyclic graph (DAG) of jobs, which can be displayed on demand.

Considerable efforts are usually spent to tweak parameters for the various programs involved in a workflow. To optimize parameter usage the params-directive is often used. One can specify data and working directories as well as parameters used in the execution of the rule. Parameters can be specified explicitly within a rule, but it is considered best practice to define all parameters in the config file. To start the workflow via a configuration file, the –configfile command-line argument with the name of the configuration file is required. Log files are used to monitor the pipeline process and to look at processes after finished execution. In addition, Snakemake directly supports benchmark testing via the benchmark-directive which allows to capture the CPU load and the memory usage and gather runtime information. Furthermore, the management of software is directly integrated into Snakemake. Each rule in Snakmake can use its own predefined environment using Conda integration, which allows it to automatically download and run dependencies. Each Conda environment is isolated and can be configured in a YAML file. Instead of Conda environments, it is also possible to define Docker [30] and Singularity [38] containers for each rule. To make widely used tools easily accessible to users, Snakemake has a central public repository (https://snakemake-wrappers.readthedocs.io) that allows the scientific community to share wrappers with each other. A tool wrapper provides the workflow with a Python or R script that either uses libraries of the respective scripting language or calls a shell command. Each wrapper creates an individual Conda environment with

the required tool and library version moreover it solves conflicts among the dependencies. Snakemake allows to adjust the execution to run on different computational environments such as HPC clusters, cloud computing resources as offered by Amazon Web Services [34] and similar providers, or locally on the system.

3.4.1 Implementation

As part of this thesis, we translated the SmRNASeq workflow into Snakemake. Figure 5 shows the architecture of the newly built workflow in form of a directed acyclic graph which was generated by Snakemake. Nodes of the DAG represent jobs and a directed edge indicates that the targeted job needs the output of the other job as input. A sequence of jobs that depend on each other is presented as a path. Therefore, two disjoint paths can be executed in parallel.
During the implementation of the SmRNASeq workflow, , we stayed as close as possible to the workflow code to avoid differences and to make further work on the project easier. In SmRNASeq, all rules were defined in one file, main.nf. To improve the readability of the code, we instead used several files in our implementation. SmRNASeq uses Charliecloud [19] container for the software management, which deploy Conda [18] environments. In our implementation each rule has its own Conda environment, in which its required packages are managed and checked for compatibility, to ensure portability. Since Conda environments are used, the shell directive had to be used in the Snakemake workflow. When running the Nextflow pipeline, parameters for the input files and instructions for the tools are passed via the command line. This was generalized and simplified in Snakemake via a configuration file. Furthermore, to ensure that both workflows use the same software versions and are executed with the same parameters, it was decided not to use Snakemake wrappers and their benefits. The SmRNASeq workflow defines the default resources allocated to each different step in the workflow in the base config. If an attempt fails with an exit code that corresponds to a lack of resources, the workflow automatically increases the resources up to the specified maximum. The Snakemake workflow allocates resources in each rule and also increases resources if an attempt fails. The maximum is defined by the command line argument *–cores*. Moreover, to create the same result folder structure, low-level code was added when a tool did not have a standalone output directory command.
Our workflow can be found on and downloaded from Gitlab. A comprehensive explanation of how to install and run the pipeline is available there. Direct link to the release: https://gitlab.cs.uni-duesseldorf.de/albi/albi-students/ba-florian-radziej

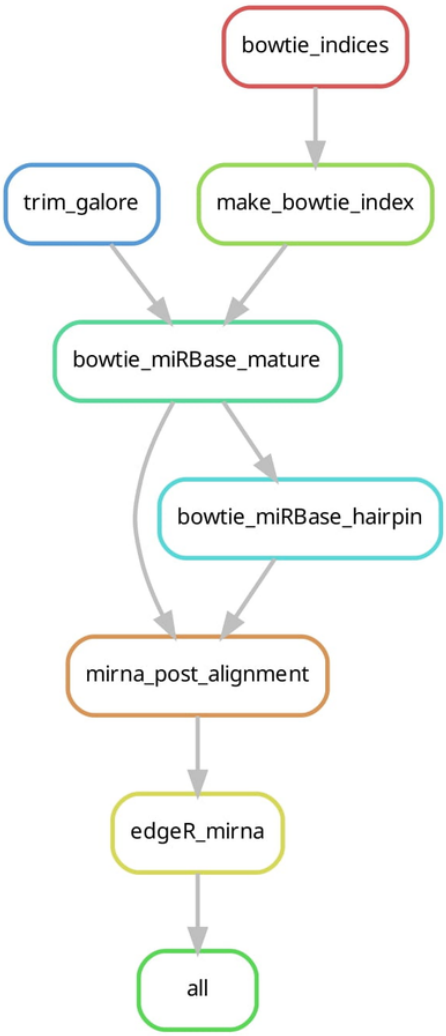

Figure 5: Directed Acyclic Graph of the self-created pipeline in Snakemake

4 Results

In this section, we compare both pipelines with each other. We examine the runtime of the workflows and highlight the main similarities and differences. For this purpose, both workflows were applied to the same sequencing and reference data, which yielded identical analysis results. When executing the workflows, particular emphasis was taken to ensure that the parameters within the process, as well as the resource allocation, were identical.

4.1 Data

4.1.1 Sequencing Data

For this thesis, the Institute of Pathology of the UKD [2] provided us with a miRNA sequencing data set as shown in Table 1. The set consists of 29 different samples in the compressed FastQ format. As mentioned in Section 2.2, the IPMN samples are categorized by their histological characteristics. In Table 1, the colors represent the degree of dysplasia. Normal tissue has the lowest level of dysplasia, while tumor tissue has the highest level. The abbreviations for the different types of samples are N, T and P for normal, tumor and pancreaticobiliary IMPNs, respectively. Furthermore, intestinal and gastric IMPNs are distinguished into low and high grade which results in the abbreviations i_HG, i_LG, g_HG and g_LG.

Tissue Type	Sample Count
normal tissue	4
intestinal IPMN low grade	4
intestinal IPMN high grade	4
gastric IPMN low grade	5
gastric IPMN high grade	4
pancreatobiliary IPMN	3
tumor tissue	5

Table 1: Counts of samples available for each tissue type

4.1.2 Reference Data

The resource for the reference miRNA data is a public repository called miRBase [39]. miRBase provides a wide range of miRNA data including sequences, biogenesis precursors, genome coordinates and context for each sample. We have access to 38589 entries representing hairpin precursor microRNAs, from 271 organisms as well as the resulting 48860 different mature miRNA sequences. Before a miRNA is published, names are assigned under a standard nomenclature system [40]. The species of origin is designated with a three-letter prefix, for example the shortcut hsa stands for homo sapiens. The prefix "miR-" refers to the mature form of the miRNA, while the uncapitalized "mir-" refers to the hairpin miRNA. Every miRNA has a specific numeric suffix. If miRNAs have identical sequences expect of one or two nucleotides, an

additional lower case letter is added. The structural layout of miRNA sequence data can be useful to understand the processes in the workflow. [39]

4.2 Runtime

The runtime of both workflows is examined in the following. Benchmark tests for both pipelines were performed on the same computer provided by the chair of Algorithmic Bioinformatics, which has an AMD EPYC 7742 64-Core Processor with 128 Threads and 1 TiB DDR4 RAM. The sequencing and reference data mentioned in Section 4.1.1 and 4.1.2 were used to run the workflows. To make sure that the runtimes are comparable and not affected by differences in their configurations of the pipelines themselves, we worked with command line arguments and adjusted the configuration files.
In the SmRNASeq pipeline, changes were made to *nextflow.config*. To ensure local execution and maximum usage of the available CPUs on the computer, we added the executor *local* and the executor attribute *cpus*, which allocates the number of CPUs, to the Charliecloud profile. Furthermore, the maximum usage of CPUs per job request was limited via the command line argument **–max_cpus**. In the Snakemake execution, we passed two command line arguments to create an identical configuration. Snakemake ensures that the total number of CPUs consumed by a workflow does not exceed the total number of cores defined by **–cores** and the respective parameter. The command line argument **–max-threads** sets a global maximum number of CPUs for any job. All benchmarks were performed with the built-in bash command *time* [41], which determines how long an executed command takes to run. The running times were measured with and without previously installed environment management systems of the respective workflow. Each run with its respective parameters was repeated three times and the resulting runtimes were averaged.

Pipeline	Environments Installed	Maximum CPUs	CPUs per Task	Runtime (H:MM:SS)
Snakemake	Yes	128	16	0:02:11
Snakemake	No	128	16	0:02:42
Nextflow	Yes	128	16	0:02:06
Nextflow	No	128	16	0:02:54

Table 2: Runtimes with default settings of SmRNASeq

Pipeline	Environments Installed	Maximum CPUs	CPUs per Task	Runtime (H:MM:SS)
Snakemake	Yes	128	1	0:03:58
Nextflow	Yes	128	1	0:04:23
Snakemake	Yes	16	1	0:06:34
Nextflow	Yes	16	1	0:05:31

Table 3: Runtimes with one CPU per task and different allocated maximum CPUs

We tested both pipelines with the default settings of the SmRNASeq workflow, which runs with a maximum of 128 allocated CPUs and a limit of 16 CPUs per task. No processes were restricted by this limit, since only jobs with a maximum of 6 CPUs were executed. As shown in Table 2 the workflows with previously installed environments are faster. The difference is caused by setting up the environments which includes downloading and installing the dependencies as well as resolving conflicts within the dependencies. During the tests, we noticed a difference in runtime when we limited the maximum CPUs per task. To get a broader picture, we limited the maximum number of CPUs per task to 1 CPU. We also limited the maximum allocated CPUs to see runtimes with limited amount of parallel running jobs. The execution was restricted to 16 core with a maximum of one core per task as shown in Table 3. As expected, the workflows with restricted CPUs per task need more time than the default configuration with 16 CPUs per task. The execution with default configuration of both workflows need around 2 minutes while the workflows with similar allocated CPUs, but restricted CPUs per task take about twice the length of time. Furthermore, as expected, the most time-consuming workflows are those with a maximum of 16 CPUs, which take 5.5 minutes and 6.5 minutes, respectively. We have noticed that with a maximum allocated cores of 128 and a limit of 1 CPU per task Snakemake is approximately 20 seconds faster than Nextflow, but in each other case Nextflow is faster than Snakemake. We discussed with Nextflow developers in the nf-core community, and we suggest that differences in the way Snakemake and Nextflow are orchestrating the pipeline lead to the differences in the runtimes.
The team around Elise Larsonneur [42] also tested various workflow management systems for their runtime. They built their own workflow to test specific features of workflow management systems, as well as scripts to measure certain parameters. The presented study was performed locally on a single computation node. For the constructed pipeline from the study, Nextflow took 4.0 minutes to execute. Snakemake was slightly faster at 3.7 minutes.

4.3 Comparison

To structure the comparison, the features of the pipeline tools Snakemake and Nextflow were classified into the categories setup, philosophy of workflow execution, and features.

4.3.1 Setup

The documentation of both pipeline tools is well structured, extensive and clear. To help beginners with the first steps, both tools have a very detailed tutorial.
Nextflow and Snakemake use domain specific language (DSL) extensions of Groovy and Python, respectively. Python is a well-known language [43] that we think makes Snakemake potentially easier to learn and to share between colleagues. Groovy is less popular [43], but for moderately experienced programmers easy to pick up. A benefit in both cases is the possibility to use the underlying languages beyond the domain specification as required.
Both tools are supported by a large community. For lack of better measures, judging by

stars in the respective Github repository and number of forks, Nextflow is more popular and more widely used than Snakemake. An advantage of Nextflow is the active user community project nf-core [3]. The community collects curated sets of best-practice bioinformatics analysis pipelines that can run on most computing infrastructures. Questions about specific pipelines or Nextflow can be asked via the instant messenger service Slack, which has an active community support from developers and experienced Nextflow users. The Snakemake community is still in the early stage of collecting and maintaining finished pipelines in Snakemake. (https://github.com/snakemake-workflows)
The advantage of a large active community as well as pipeline templates and extensive documentation from Nextflow is very valuable in our opinion. Groovy may be an uncommon choice for a pipeline, but due to the DSL layout from Nextflow Groovy is just used for difficult cases.

4.3.2 Philosophy of Workflow Execution

A major difference between Snakemake and Nextflow is the approach to how the hierarchy of workflow tasks is built after execution. Both workflows follow a different philosophy. Snakemake adopts the approach from the build system Make [44] that has inspired its design. The philosophy behind the approach can be described as "bottom-up", meaning inputs and outputs are defined and a specific final output is specified (usually defined in the all rule). Snakemake builds all dependencies to other rules recursively starting from the final files. This architecture allows the dry-run feature of Snakemake. By beginning to build the hierarchy of tasks from the output files, Snakemake can check if input files exist and display commands to be performed without executing them. Dry-runs are helpful in testing input data and debugging processes.
Nextflow follows the rather natural philosophy of the "top-down" principle, better known as the dataflow model. Nextflow starts processes and runs them, if the input is available, and waits, if there is no output from the input channel yet. The order of execution of the processes is determined by the channels, implicit by input and output declaration. Contrary to Snakemake, there is no explicit final output to which all processes are oriented, but the order depends on the channels. A dry-run feature is not implemented, since Nextflow does not know which files are created before the run. The best way to get a similar debugging option is to use small test data set to emulate the procedure of a normal execution.
Nextflow uses channels to communicate between tasks. Channels can manipulate and change data to provide more flexibility. Moreover, Nextflow is not limited to files, but can use any data structure and does not need to specify the exact number of files to output. This distinguishes it from Snakemake, where only files can be used. Since in Snakemake each file must be explicitly specified, wildcards are used to generalize the file names.
Snakemake interprets all paths of files in relation to the folder where the execution was performed or a explicit specified directory. This is necessary to ensure a unique file scheme to prevent result files from overwriting each other. In order to sort and move files to other directories, low-level code is needed in the respective tasks. Nextflow has a different work directory

abstraction. Each individual process is executed in its own directory, thus files cannot overwrite each other. These directories are in the work folder and not supposed to be considered by the user. Instead, if a process is annotated with the *publishDir* directive, a directory can be specified into which the output files are copied. Only files that match the declaration of the output directive will be published in the result folder.
In our experience, the dataflow model Nextflow is based on, is more intuitive than the Snakemake approach, especially for beginners. Furthermore, Nextflow offers more options to modify working and result directories. In contrast to Nextflow, the structuring of results via low-level code in Snakemake is not elegant, and we would prefer to use *publishDir*. This outweighs the advantage of a dry-run mode in Snakemake.

4.3.3 Features

Snakemake and Nextflow provide features, which partly differ in their options and functionality. This section evaluates features that were used in the work on the thesis or which seemed worth mentioning.
Programs sometimes create files that are not needed as input for the next step or are not of particular interest to the user, therefore they should not appear in the result directory. Snakemake allows to specify output files with a *temp* annotation, which are then deleted after all input rules that use these output files are completed. This functionality is not available in Nextflow, but it is possible to modify the output options. Result files can be explicitly specified via *publishDir* directive.
Snakemake as well as Nextflow have the option to send notifications by mail when the workflow is completed. In Snakemake, only the Unix command *mail* [45] can be included in the workflow. This will send a simple mail with a notification specified by the user. Nextflow provides, in comparison more advanced mail functions. Notifications can be send automatically when a workflow is executed with the - *N* option. This option sends a pre-designed message to the recipient address. Alternatively mails can be individually created and configured in the workflow.
Both, Nextflow and Snakemake can automatically determine which tasks need to be rerun when parameters, code, or input files have changed. Snakemake will reuse the input files that are found in the storage per default. Other than Nextflow, where a rerun needs to be specified with the parameter *-resume*. Nextflow will use cached results where the inputs are the same.
To better understand the structure of a workflow, visualizations are useful. Both pipelines offer the possibility to display their respective directed acyclic graph on demand.
Furthermore, both pipelines have integrated software management systems. Snakemake can use the Conda package manager and supports Docker and Singularity container to manage software. Nextflow has integrated the aforementioned options and alternatively offers Charliecloud, Podman and Shifter containers.
Snakemake can make use of cluster engines such as SLURM and SGE and cloud computing

resources as offered by Amazon Web Services, Google Life Sciences, GA4GH TES as well as Kubernetes. [5] Nextflow furthermore provides Azure Batch and multiple resource manager like PBS Pro and OAR. [4] Since we id not use them, only a theoretical assessment can be made. Nextflow's configuration of computational environments looks simpler and more individual, since single processes can be executed differently. For example one process can be executed locally, while the others can run with a cluster execution. This can be achieved with the change of a single executor parameter.

Working with both pipelines for this thesis, our impression was that Nextflow is far richer in the scope of features and more diverse in options to customize the workflow. Nextflow offers a broader range of software management systems and computational environments, which provide more variety and flexibility to make a more appropriate choice for the specific workflow. The functions of Nextflow felt more powerful and extensive than those of Snakemake.

5 Discussion

We compared the workflow management systems Nextflow and Snakemake using the curated pipeline SmRNASeq. We will now discuss the results. First, we looked at the runtimes of both workflows. Nextflow was faster than Snakemake in the default configuration, but the difference of around 5 seconds is not relevant to a user. In the paper on Evaluating Workflow Management Systems [42], Snakemake was slightly faster. The reasons for the differences in runtime could not be identified in the course of the thesis. We assume that the time difference is related to the orchestration of the (improved) pipelines. Since both workflow management projects were unknown to us before this thesis, we started the comparison unbiased.During the work period, we wrote code in Snakemake, but only read Nextflow files. Due to the different ways of working with the pipelines, our perception may be distorted. Snakemake and Nextflow are both adequate tools to create pipelines. The choice of a workflow management system depends on personal preferences, for example concerning the philosophy of workflow execution, choice of the underlying language, the assistance with problems and how a pipeline can be shared with colleagues. Should these factors not yield any compelling arguments for either system, we recommend working with Nextflow. The workflow execution approach is more intuitive for beginners, compared to the Snakemake "bottom-up" approach which is quite confusing at first. Nextflow allows a more simple scale up for more computationally expensive pipelines. The Nextflow community provides a comprehensive platform to support users as well as providing curated pipelines. The direct comparison of Snakemake and Nextflow left us with the impression that Nextflow offers significantly more features and capabilities which outweigh its lack of a dry-run mode and the use of Groovy. We have also expressed our opinion on various features in the Section 4.3.

6 Outlook

The implementation of SmRNASeq in Snakemake is not yet complete. As part of this work, we only translated the pathway marked in green in Figure 4. With the groundwork laid in this thesis, a full conversion of the pipeline can be achieved by translating the remaining quality control processes, the compilation of the results, and the identification of new and known miRNAs.

The execution of the Snakemake workflow could also be improved by building a more user-friendly summary report. Such a report could provide an overview of parameters, and information about input and meta data used during the execution of the workflow. The report could be printed in the terminal similar to the SmRNASeq workflow. Furthermore, Snakemake provides a report function that automatically generates self-contained HTML reports that collect results, runtime statistics and workflow specific information, which could also be implemented.

Both workflows consist of a relatively small number of seven tasks, eight if you count the rule *all* in the Snakemake workflow. The workflows take about 2 minutes at maximum utilization

of the CPUs of the computer that we performed the benchmark test on. The workflows and runtimes are too small to measure siginifcate differences. A pipeline with a larger number of tasks would be interesting to investigate to see whether the elapsed workflow runtime increases significantly with the number of tasks. With this it could be possible to find differences in the parallelization management or time requirement to build the job hierarchy.
The paper on Evaluating Workflow Management Systems [42] mentioned in Section 4.2 examines more metrics for evaluating the efficiency of pipelines. Additional benchmark tests, for example on CPU load or memory consumption, could extend the comparison between the pipelines and provide additional insights. Furthermore, rerunning the pipeline of the paper with the current Nextflow and Snakemake versions and comparing the benchmarking results to the ones in the paper would be interesting since the paper is from 2018. In conjunction with the results of this thesis, which measured a faster runtime for Nextflow for the specific SmRNASeq workflow in the default settings, while in the paper Snakemake was quicker. It could be determined whether the difference between paper and thesis is due to a new orchestration or resource allocation of the improved workflow management systems.
The SmRNASeq pipeline received a major structural update during work period of this thesis. The pipeline was ported to the updated Nextflow DSL2 syntax which includes comprehensive improvements through modularization, encapsulated dependencies, and improved data flow manipulation. A comparison could be made with the new version of the workflow.

References

[1] Jacob O'Brien et al. "Overview of MicroRNA Biogenesis, Mechanisms of Actions, and Circulation". In: *Frontiers in Endocrinology* 9 (2018). ISSN: 1664-2392.

[2] *Institut für Pathologie*. https://www.uniklinik-duesseldorf.de/patienten-besucher/klinikeninstitutezentren/institut-fuer-pathologie. June 2022.

[3] Phil Ewels et al. *Nf-Core/Smrnaseq: V.1.1.0 - 2021-05-11*. Zenodo. June 2021. DOI: 10.5281/zenodo.4956678.

[4] Paolo Di Tommaso et al. "Nextflow Enables Reproducible Computational Workflows". In: *Nature Biotechnology* 35.4 (Apr. 2017), pp. 316–319. ISSN: 1546-1696. DOI: 10.1038/nbt.3820.

[5] Felix Mölder et al. "Sustainable Data Analysis with Snakemake". In: *F1000Research* 10 (Jan. 2021), p. 33. ISSN: 2046-1402. DOI: 10.12688/f1000research.29032.1.

[6] Vincenzo Nasca et al. "Intraductal Pancreatic Mucinous Neoplasms: A Tumor-Biology Based Approach for Risk Stratification". In: *International Journal of Molecular Sciences* 21.17 (Jan. 2020), p. 6386. ISSN: 1422-0067. DOI: 10.3390/ijms21176386.

[7] Scott M. Hammond. "An Overview of microRNAs". In: *Advanced Drug Delivery Reviews* 87 (June 2015), pp. 3–14. ISSN: 0169409X. DOI: 10.1016/j.addr.2015.05.001.

[8] *Cancer Statistics, 2020 - Siegel - 2020 - CA: A Cancer Journal for Clinicians - Wiley Online Library*. https://acsjournals.onlinelibrary.wiley.com/doi/10.3322/caac.21590.

[9] Kyle Strimbu and Jorge A. Tavel. "What Are Biomarkers?" In: *Current opinion in HIV and AIDS* 5.6 (Nov. 2010), pp. 463–466. ISSN: 1746-630X. DOI: 10.1097/COH.0b013e32833ed177.

[10] Ricardo Rocha et al. "Spontaneous Rupture of Pancreatic Pseudocyst: Report of Two Cases". In: *Case Reports in Surgery* 2016 (2016), pp. 1–3. ISSN: 2090-6900, 2090-6919. DOI: 10.1155/2016/7056567.

[11] Sarka Benesova, Mikael Kubista, and Lukas Valihrach. "Small RNA-Sequencing: Approaches and Considerations for miRNA Analysis". In: *Diagnostics* 11.6 (June 2021), p. 964. ISSN: 2075-4418. DOI: 10.3390/diagnostics11060964.

[12] *An Introduction to Next-Generation Sequencing Technology*. https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf.

[13] Peter J. A. Cock et al. "The Sanger FASTQ File Format for Sequences with Quality Scores, and the Solexa/Illumina FASTQ Variants". In: *Nucleic Acids Research* 38.6 (Apr. 2010), pp. 1767–1771. ISSN: 0305-1048. DOI: 10.1093/nar/gkp1137.

[14] Ana Conesa et al. "A Survey of Best Practices for RNA-seq Data Analysis". In: *Genome Biology* 17.1 (Jan. 2016), p. 13. ISSN: 1474-760X. DOI: 10.1186/s13059-016-0881-8.

[15] Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. “edgeR: A Bioconductor Package for Differential Expression Analysis of Digital Gene Expression Data”. In: *Bioinformatics (Oxford, England)* 26.1 (Jan. 2010), pp. 139–140. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp616.

[16] Michael Love et al. *DESeq2: Differential Gene Expression Analysis Based on the Negative Binomial Distribution*. Bioconductor version: Release (3.15). 2022. DOI: 10.18129/B9.bioc.DESeq2.

[17] R Core Team. *R: A Language and Environment for Statistical Computing*. Manual. R Foundation for Statistical Computing. Vienna, Austria, 2021.

[18] *Anaconda Software Distribution*. https://docs.anaconda.com/.

[19] Reid Priedhorsky and Timothy C. Randles. *Charliecloud: Unprivileged Containers for User-Defined Software Stacks in HPC*. Tech. rep. LA-UR-16-22370. Los Alamos National Lab. (LANL), Los Alamos, NM (United States), Aug. 2016. DOI: 10.2172/1296650.

[20] Björn Grüning et al. “Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences”. In: *Nature Methods* 15.7 (July 2018), pp. 475–476. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0046-7.

[21] Babraham Bioinformatics. *Babraham Bioinformatics - Trim Galore!* https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/. June 2022.

[22] Marcel Martin. “Cutadapt Removes Adapter Sequences from High-Throughput Sequencing Reads”. In: *EMBnet.journal* 17.1 (May 2011), pp. 10–12. ISSN: 2226-6089. DOI: 10.14806/ej.17.1.200.

[23] Babraham Bioinformatics. *Babraham Bioinformatics - FastQC A Quality Control Tool for High Throughput Sequence Data*. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. June 2022.

[24] Ben Langmead. “Aligning Short Sequencing Reads with Bowtie”. In: *Current Protocols in Bioinformatics* 32.1 (2010), pp. 11.7.1–11.7.14. ISSN: 1934-340X. DOI: 10.1002/0471250953.bi1107s32.

[25] Petr Danecek et al. “Twelve Years of SAMtools and BCFtools”. In: *GigaScience* 10.2 (Feb. 2021), giab008. ISSN: 2047-217X. DOI: 10.1093/gigascience/giab008.

[26] G.J. Hannon. *FASTX-Toolkit*. 2010.

[27] Wei Shen. *SeqKit - a Cross-Platform and Ultrafast Toolkit for FASTA/Q File Manipulation*. July 2022.

[28] Dierk Koenig and Andrew Glover. *Groovy in Action*. Manning, 2007.

[29] Ken Arnold, James Gosling, and David Holmes. *The Java Programming Language*. Addison Wesley Professional, 2005.

[30] Dirk Merkel. *Docker: Lightweight Linux Containers for Consistent Development and Deployment: Linux Journal: Vol 2014, No 239*. https://dl.acm.org/doi/10.5555/2600239.2600241.

[31] Gregory M. Kurtzer et al. *Hpcng/Singularity: Singularity 3.7.3*. Zenodo. Apr. 2021. DOI: 10.5281/zenodo.4667718.

[32] *Containers Organization Podman*. https://podman.io/.

[33] Douglas M Jacobsen and Richard Shane Canon. *Contain This, Unleashing Docker for HPC*.

[34] Thomas J. Leeper. *Aws.S3: AWS S3 Client Package*. 2020.

[35] Philip A. Ewels et al. "The Nf-Core Framework for Community-Curated Bioinformatics Pipelines". In: *Nature Biotechnology* 38.3 (Mar. 2020), pp. 276–278. ISSN: 1546-1696. DOI: 10.1038/s41587-020-0439-x.

[36] Marc R. Friedländer et al. "miRDeep2 Accurately Identifies Known and Hundreds of Novel microRNA Genes in Seven Animal Clades". In: *Nucleic Acids Research* 40.1 (Jan. 2012), pp. 37–52. ISSN: 0305-1048. DOI: 10.1093/nar/gkr688.

[37] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 978-1-4414-1269-0.

[38] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific Containers for Mobility of Compute". In: *PLOS ONE* 12.5 (May 2017), e0177459. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0177459.

[39] Ana Kozomara and Sam Griffiths-Jones. "miRBase: Annotating High Confidence microRNAs Using Deep Sequencing Data". In: *Nucleic Acids Research* 42.D1 (Jan. 2014), pp. D68–D73. ISSN: 0305-1048. DOI: 10.1093/nar/gkt1181.

[40] Sam Griffiths-Jones et al. "miRBase: Tools for microRNA Genomics". In: *Nucleic Acids Research* 36.suppl_1 (Jan. 2008), pp. D154–D158. ISSN: 0305-1048. DOI: 10.1093/nar/gkm952.

[41] *Time(1) - Linux Manual Page*. https://man7.org/linux/man-pages/man1/time.1.html.

[42] Elise Larsonneur et al. "Evaluating Workflow Management Systems: A Bioinformatics Use Case". In: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. Dec. 2018, pp. 2773–2775. DOI: 10.1109/BIBM.2018.8621141.

[43] *Top Computer Languages 2022 - StatisticsTimes.Com*. https://statisticstimes.com/tech/top-computer-languages.php.

[44] Richard M. Stallman and Roland McGrath. *GNU Make - A Program for Directing Recompilation*. 1991.

[45] *Stg-Mail(1) - Linux Manual Page*. https://man7.org/linux/man-pages/man1/stg-mail.1.html.