Automating RNA-seq data analysis through a computational workflow using Snakemake

Pinky Debnath[a]

a: Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability

Abstract:

RNA-sequencing (RNA-seq) is a technology to explore the RNA in a sample through high throughput sequencing methods. To quantify and analyzing the RNA in cells, RNA sequencing mostly takes the advances of the Next generation sequencing (NGS) technology. The advances include effective identification of differentially expressed genes, studying the entire transcriptome and so on. Meanwhile, RNA-seq is the one of the most potential techniques to deeply understanding the biology of cells, transcriptome and the expression of genes. The extraction of genetic information from RNA-seq data requires a large number of computational tools, with subsequent software and integration of the reference data. Regarding this, RNA-seq data analysis requisites a series of computational steps in a proper sequence, which is performed by a specialized category of workflow management system. Computational steps have been performed to prevent the repetition of steps and making the workflow more precise and effective. Another requirement for the optimized RNA-seq data analysis is that the data analysis should be reproducible and scalable. Snakemake is a useful and worthwhile workflow management system to create reproducible and reliably working data analysis pipelines. The workflow is specified with a customized Snakefile which is based on the Python standard programming language. The file consists of rules about creating output files from input files, workflow dependencies and shell command lines. Here, we have developed an automated workflow in Snakemake to analyze RNA-seq data from different species, using several Bioinformatics tools. Besides, we aimed to make the workflow more generalized and user-friendly for exploring any paired-end RNA seq data effectively and frequently through employing the workflow management system Snakemake and the Bioinformatics package management system Conda.

Keywords: RNA-seq data, Snakemake, Bioinformatics tools.

Year: Summer semester 2022
Master module: [CS0014] Research Internship Master Chemical Biotechnology
Supervisor: Prof. Dr. Dominik Grimm
Document Type: Internship Report
Date: 5 September, 2022

1. Introduction

For studying coding, non-coding RNA and transcriptome profiling, RNA-seq is an unquestionable technology. RNA-seq data analysis is efficacious for the identification of post-transcriptional modification during mRNA (messenger RNA) processing. Moreover, RNA-seq is also adequate for understanding the biology of cells, indicating toward the diseases through analyzing the level of transcription, the timeline of activation of genes and marking the genes expressed in particular cells (Ozsolak, 2011). RNA-seq is performed in several steps. The initial step is the extraction of RNA and cDNA (complimentary DNA) library preparation by reverse transcription. Then putting the samples into NGS (Next generation sequencing) workflow. After that, cDNA is fragmented and adapters are added to the end of each fragment to permit sequencing through the adapter containing functional elements. Furthermore, cDNA fragments are explored after quality checking, amplification and size selection and then sequenced into single (cDNA fragment sequenced from one end) or paired-end (cDNA fragment sequenced from both end) (Mackenzie, 2018). Compared to other traditional technologies such as, Real-time PCR or microarrays, RNA-seq analysis generate an extensive amount of data, which require a refined computational approach for inspecting. Moreover, RNA-seq data analysis is based on multiple steps using different Unix-based tools. The steps include quality control of raw reads, sequence read alignment, quantification of transcripts per million (TPM), read counting and downstream analysis of differential gene expression of RNA-seq data.

Intending to RNA-seq data inspecting, multiple command lines need to be executed in proper sequence to generate effective pipeline. For the automation of the generated pipeline and maintenance of reproducibility, workflow engines play crucial roles. For the analysis of complex data like RNA-seq data, scalability and reproducibility are indispensable. Snakemake is a workflow management engine relay on Python-based language, that provide fast and easy accessible environment for pipeline implementation. Compared to other workflow management systems that use domain specific language (Python and Groovy) such as Nextflow (Di Tommaso, 2017), Snakemake showed workflow processing time 3.7 minutes, whereas, Nextflow showed 4.0 minutes (Larsonneur, 2018). Despite of working in local environment, Snakemake can also work on the cloud platforms like Amazon Genomics CLI, Google Cloud CLI . Moreover, Snakemake allow stepwise modularization to improve the readability of data analysis workflow, with the integration of Conda environment. Conda is a package and environmental management system for finding and installation of packages related to Bioinformatics (Anaconda Software Distribution, 2020). Snakemake also provide genuine traceability and documentation by tracking the input files, output files, parameters, software and the executed code during workflow processing and generate HTML based reports after fulfillment (Mölder et al., 2021). Furthermore, automatic scalability is provided by Snakemake by optimizing the number of parallel processes (Köster, 2012). Also, Snakemake has the capability of eliminating the duplicate tasks for the same sample (Mölder et al., 2021).

Therefore, the overall goal of the project is to develop a fully reproducible Snakemake workflow for analyzing RNA-seq data. More specifically to inspect the different bioinformatics tools namely, `FastQC` (Andrews, 2010), `MultiQC` (Ewels et al., 2016), `Cutadapt` (Magoč and Salzberg, 2011), `Kallisto` (Bray et al., 2016), `Salmon` (Patro et al., 2017) , `STAR indexing and mapping` (Dobin et al., 2013), `FeatureCounts` (Liao et al., 2014), `HTSeq-count` (Anders et al., 2014) `and HTSeq-qa` by generating pipeline in Snakemake workflow system. Finding the correlation and comparison within alignment independent (Kallisto and Salmon) and alignment dependent (FeatureCounts and HTSeq-count) tools, with the aim of finding out the efficiency.

In the following chapter, we have discussed about the overall workflow generation method through Snakemake in order to analyze the paired-end RNA-seq data. At the very beginning, we have mentioned about basic RNA-seq data analysis, brief description about implemented data, what is the Snakemake workflow engine and how does it work. Moreover, the discussion about the Bioinformatics tools applied to the workflow development, their functionalities and the execution procedure in Snakemake. In the meantime, we have learned about the Transcript Per Million (TPM) value and the TPM calculation technique. Eventually, we provide a best-practice example of comprehensive Snakemake workflow for paired-end RNA-Seq data analysis from beginning to end, including all necessary tools and steps on GitHub: **https://github.com/Pinky-cloud224/snakemake_workflow_rna_seq_data_analysis**

2. Method:

For the RNA-seq data analysis, at first we have to import the raw RNA-seq reads, after that we have to check the data quality through FastQC, which is based on the per base sequence content and per sequence GC content. After that, MultiQC report has to be generated including sequence counts, sequence quality histograms, per sequence quality scores, per base sequence content, per sequence GC content, per base N content, sequence length distribution, sequence duplication levels, adapter content, overrepresented sequences and FastQC status checking. To search and remove the adapters, Cutadapt has to be implemented. Next, we have to quantify the transcripts of the trimmed RNA-seq data by the alignment independent tools namely, Kallisto and Salmon. Furthermore, alignment dependent tools, for instance, STAR mapping has been carried out for quantification of gene. By using the STAR mapped BAM file we have done FeatureCounts, HTSeq-count and HTSeq quality assessment for read count and analysis.

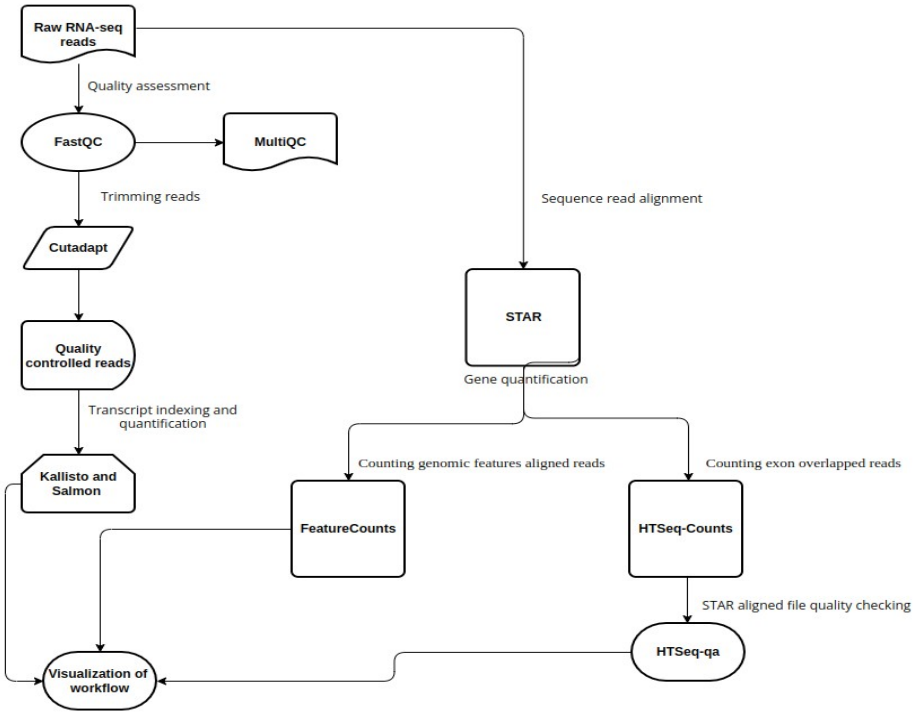


Fig. 1 Overview of the steps performed for RNA-Seq data Analysis in Snakemake Workflow.

Here, we have described the overall workflow generation method for performing automated RNA-seq data analysis in Snakemake.

2.1 Data collection: Norway rat (*Rattus norvegicus*) and human (*Homo sapiens*) genes had used to develop the workflow. Paired-end *Rattus norvegicus* (HSC_T6_S3) was the primary sample gene to create the pipeline and *Homo sapiens* (LX2_S1_R1) had been appointed to test the efficiency of the pipeline.

2.2 Snakemake workflow: Snakemake (Mölder et al., 2021) is a Python based workflow management tool. In Snakemake the entire workflow had distributed into small steps which were defined by the *rules* with particular names. In regard to execute the command lines in Snakemake, "Snakefile" had prepared, including *inputs* (the rules interpret by the dependencies), *outputs* (the rules interpret by the targets) and the scripts defined in *shell blocks*. The *input, output* and *shell keywords* were followed by a colon. The Snakefile can be generated with any types of choice of editor. The file named "Snakefile" obligates to start with a capital S without any file extension. To avoid the repetitions of typing the same pattern fastq file name in Snakemake, *wildcards* (for example: {sample}) had been used. Wildcards returned the sample or multiple samples at a time. In the shell segment, elements of the rules expressed via braces notation (also defined as placeholders), for example, expression of output, input and params in rules by *{output}, {input}* and *{params}*. The supplementary parameters correlated to the wildcards had been specified by *Params.* In order to speed up the computation, *threads* specified in the rules of the workflow. Regarding workflow execution, number of cores specified. Snakemake version 7.3.2 had used here. In snakemake, dependencies between the rules determined automatically.

Glob wildcards and Config file First of all, in the pipeline we had imported glob to generate *glob wildcard*, that would search for the filename matches to the certain filename pattern. In the meantime the *configuration file (config file)* had added in JSON or YAML format to make the customized Snakemake workflow. Configuration file added to the Snakefile via the *config* object. Sometimes it is necessary to mention the sample directory location. For example, STAR alignment require exact sample location to perform the mapping properly.

Rule all To execute the desired outcome in Snakemake, each and every steps have to be provided in snakefile, which are distributed in separate and specific rules. Moreover, Snakemake need target files to accomplish given workflow. Snakemake accepts rule names without having wildcards if there is a single rule to enforce. But if we want to carry out multiple rules at a time, Snakemake will only read the first rule as a target. For that reason, we had to mention *rule all* at the beginning of the pipeline, which contained all of the expected target files as input. To set the wildcard properly *Expand* was used.

Setting up Conda environment and Bioinformatics tool-kits By providing a .yaml extended file (env.yaml) in Snakemake workflow, C*onda environment* does need to be created. To ensure the reproducibility of the workflow the env.yaml contain all of the necessary Bioinformatics tool-kits and dependencies with specific version.

FastQC analysis The bioinformatics tool *FastQC* had been used for the quality control check of the sample SRR files. We specified the rule for FastQC analysis firstly, such as rule fastqc_analysis. FastQC generated output as fastqc.html and fastqc.zip, which had enumerated in the *Output* section including wildcard. Both output HTML and zip files appeared in the same directory as of the input file. In shell block fastQC is needed to be mentioned.

MultiQC analysis This modular tool had been used to accumulate all of the Bioinformatics results of the provided samples in a single report. For *MultiQC* analysis in Snakemake pipeline, we used the generated fastQC HTML report in *Input,* specified by multiQC inspection rule. In *Output* section, multiQC HTML report and the specified directory for multiqc report had to be named. Moreover, the shell block must include multiQC connected through Conda channel.

Cutadapt Cutadapt had implemented searching and removing the adapters. At first the adapter sequence was generated by making a sequence adapter rule and then the reverse complement sequence computed using fastx-toolkit by reverse complement computing rule. *Fastx-toolkit* is a collection of command line tools implemented for preprocessing of fastq

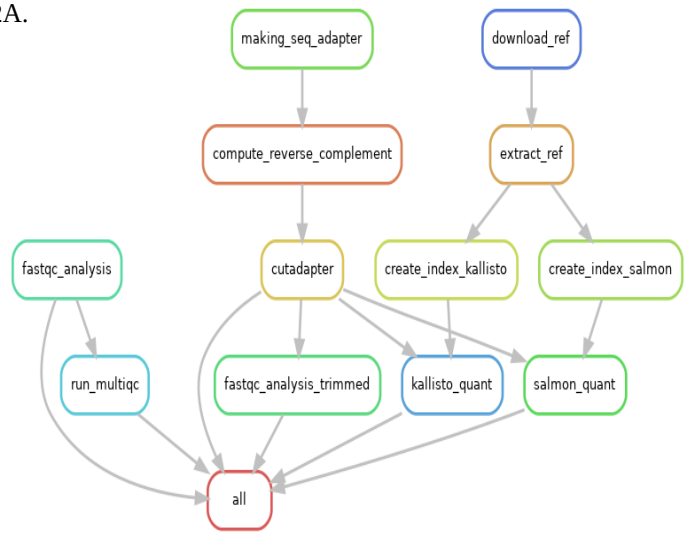
files. To keep the 5’ and 3’ ends oriented properly reverse complement sequence computed. The adapter sequence and reverse complement sequence downloaded as temporary output, that was removed later by Snakemake. The adapter was removed by adding *Cutadapt* in the Conda environment through the shell section. To remove adapter in the given paired-end reads Cutadapt used -a option for the first read and -A option for the second read. The trimming through Cutadapt must be defined by particular rule, such as rule cutadapt.

Quantification of transcripts through Kallisto and Salmon In response to the quantification of transcripts, *Kallisto* and *Salmon* are one of the most reliable and faster tools. The transcripts quantification performed in two steps (indexing and quantification) for Kallisto and for Salmon in three steps (creating directory, indexing and quantification), which had stated by definite rules for each step. The directory had to be created for Salmon, because the Salmon Index generated as a directory of files. In both of the Bioinformatics tool for indexing the *Input* section must include the reference fasta file and the output file must be mentioned with .idx extension in *Output* section. The reference fasta had to be downloaded firstly, then needed to be extracted, as the reference fasta file downloaded in zip file format. Regarding quantification the output file had to be abundance.tsv for Kallisto and quant.sf for Salmon and must be mentioned in the *Output* section with the stated output directory. The *Input* section included Cutadapt trimmed fastq files for transcripts quantification regarding both Kallisto and Salmon. Through shell block Kallisto and Salmon needed to be connected with Conda. For Kallisto and Salmon analysis, we directly included the value of *sample* wildcard in the shell command through placeholder {wildcards.sample}, because kallisto quant and salmon quant can only take the output directory names.

Snakemake workflow visualization Snakemake workflow can be represented in filegraph, rulegraph and DAG (direct acrylic graph). Filegraph mentions each and every detail, including all input and output files. Rulegraph displays only the correlation between the rules. DAG graph presents the dependencies and the rules for each job. The path represents the sequence of jobs executed consecutively. The nodes constitute for execution of rules. In case of the rules with wildcards, the value of the wildcard for the certain job is exhibited in the job node. Visualization of the Snakemake workflow is done using the dot command provided by Graphviz (Ellson, J., 2001).

The overall workflow from FastQC analysis to the Salmon and Kallisto analysis in *Rattus norvegicus* is represented visually below in rulegraph, DAG (direct acrylic graph) and filegraph:

2A.



2B.

2C.

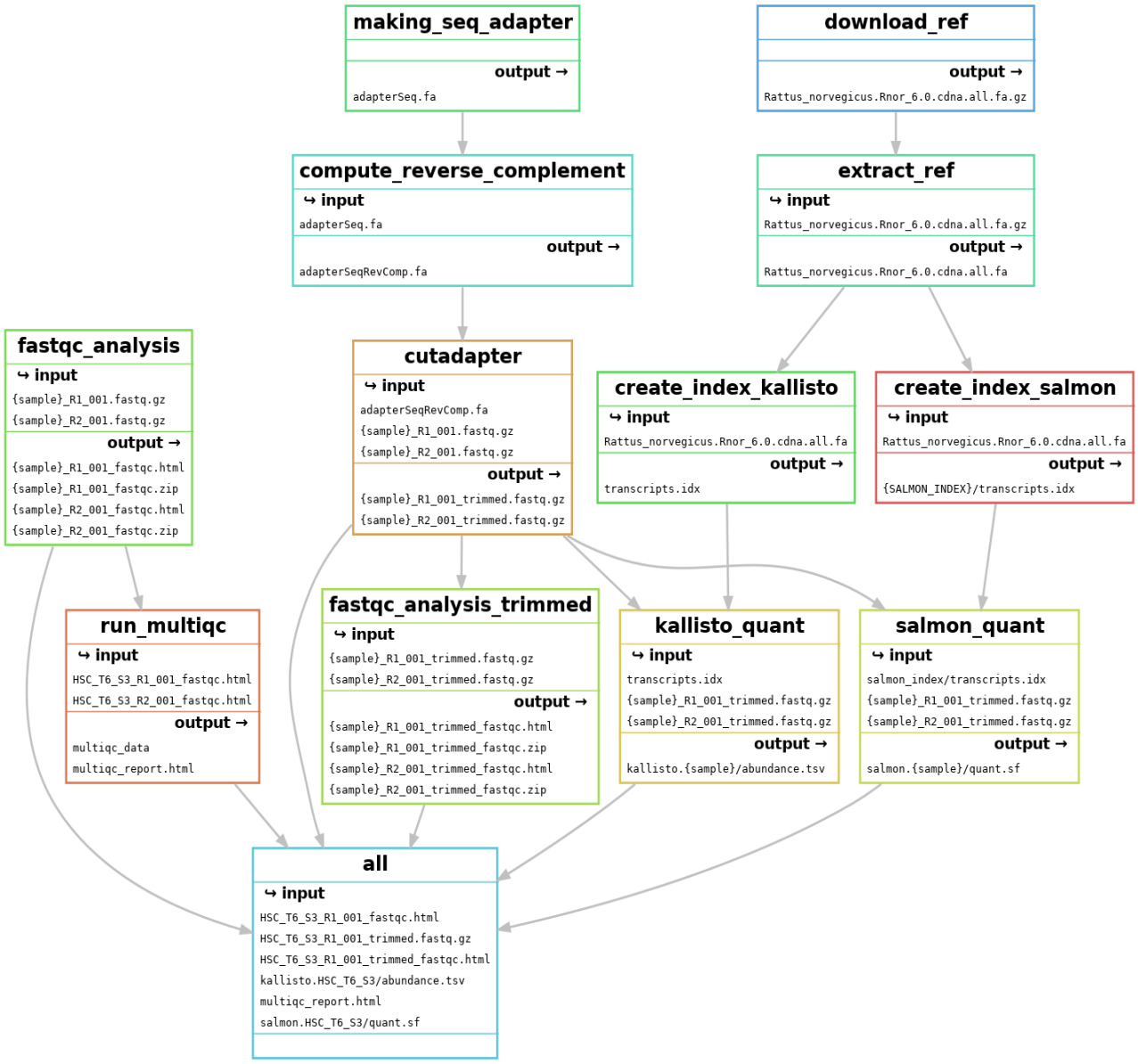


Fig. 2 Representation of Snakemake workflow of FastQC, MultiQC, Cutadapt, Kallisto and Salmon in rulegraph (2A), DAG (2B) and filegraph (2C).

STAR indexing and mapping STAR referred to as Spliced Transcripts Alignment to a Reference, is a read mapper for RNA sequencing data. STAR mapping performed in two steps, including STAR indexing and STAR read mapping. The first step of the star alignment was the indexing of the reference genome. For indexing, reference fasta and GTF files needed to be specified in the *Input* section by STAR indexing rule. The shell block for indexing consist of command lines connecting STAR through Conda. Moreover, --sjdbOverhang and –sjdbGTFfile, were used to concatenate bases from the junctions and annotations of GTF file. The output consisted of a directory with stored genome indices with *directory ()* declaration in the *Output* segment. The second step was to mapping the reads. Here, two pass mapping had been done. The 1st pass was to getting alignment and splice junction information specified by pass1 rule, then 2nd pass of the mapping was done by using the splice junction information, that was generated from the 1st pass of the mapping specified by pass2 rule. The read counting was performing at the same time. *Params* section defined the output directory where the pass-1 and pass-2

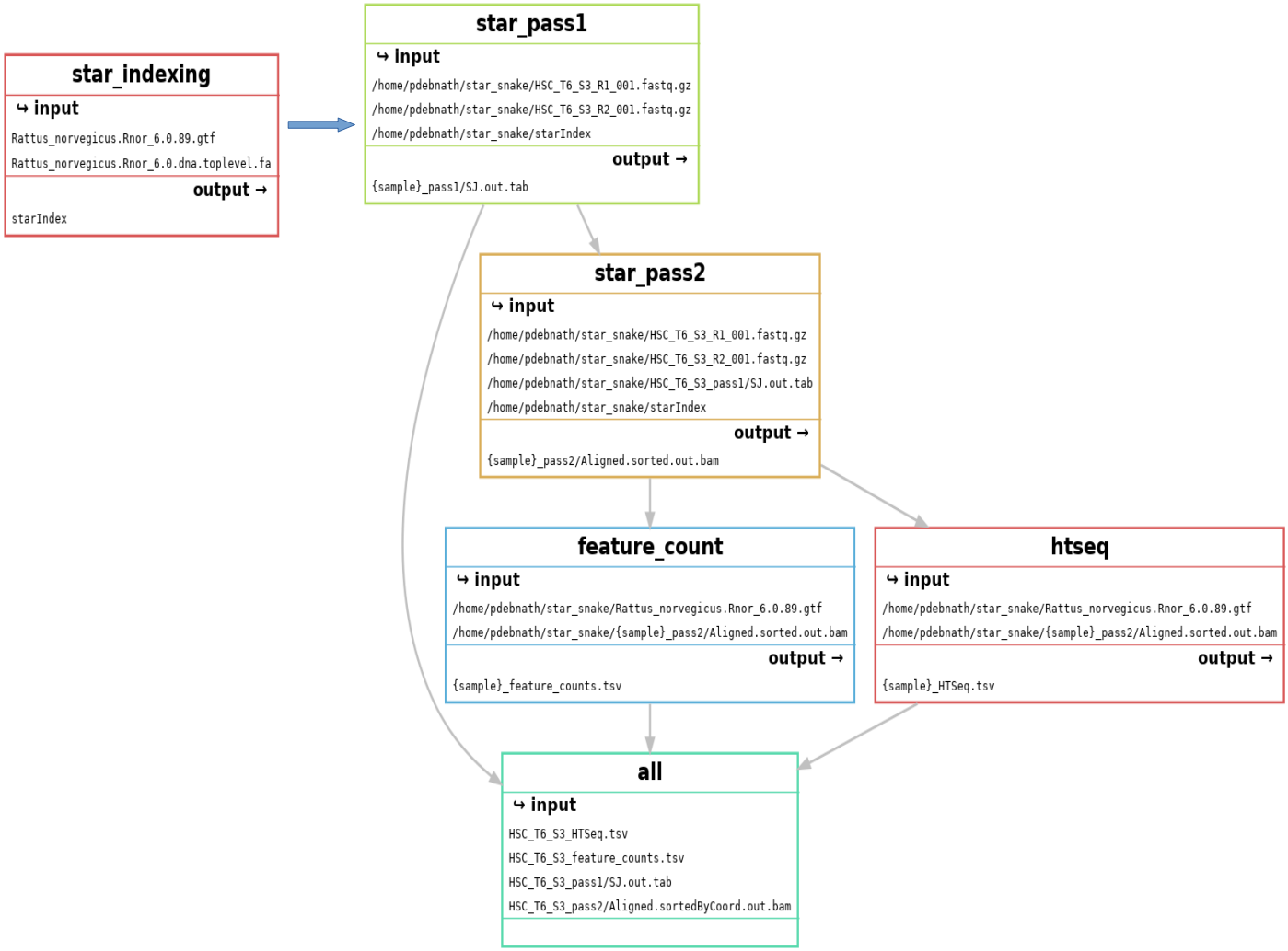
mapping results had been stored with sample ID. The shell block must contain *placeholders*, the path of the output directory where the indexed and mapped files had stored. The number of reads per gene counted here by --quantMode GeneCounts. The thread numbers obligated to be set down in *threads* segment within STAR indexing and mapping rules. The output of the STAR mapping is the binary aligned file (BAM file).

FeatureCounts In order to count the number of reads align to the genomic features, FeaturesCounts had used. The STAR aligned BAM file and GTF file used as input and defined in the *Input* section. The FeatureCounts workflow had specified by featureCounts rule. The input file root directory required to be mentioned in SAMPLE DIRECTORY. The FeatureCounts output consists of a TSV or CSV file, including chromosome number, gene length and gene counts with gene ID. The output file needed to be specified in *Output* segment of featureCounts rule with proper TSV or CSV extension.

Htseq-count To find the number of reads overlapping it's exons, HTSeq-count executed. The HTSeq-count in Snakemake workflow was defined by rule htseq. The input file had been specified by STAR aligned BAM and annotated GTF files with designated root directories in *Input* segment. Samtools view was used to convert the aligned BAM file (binary file) into the SAM file (text based). For HTSeq-count, gene ID and exon had attributed. Except of *Input*, *Output* and *Shell block*, *Thread* section was essential to be specified with particular thread numbers more than 1.

The overall snakemake workflow from STAR indexing to the HTSeq-count in *Rattus norvegicus* has been visualized beneath in filegraph, rulegraph and DAG:

3A.



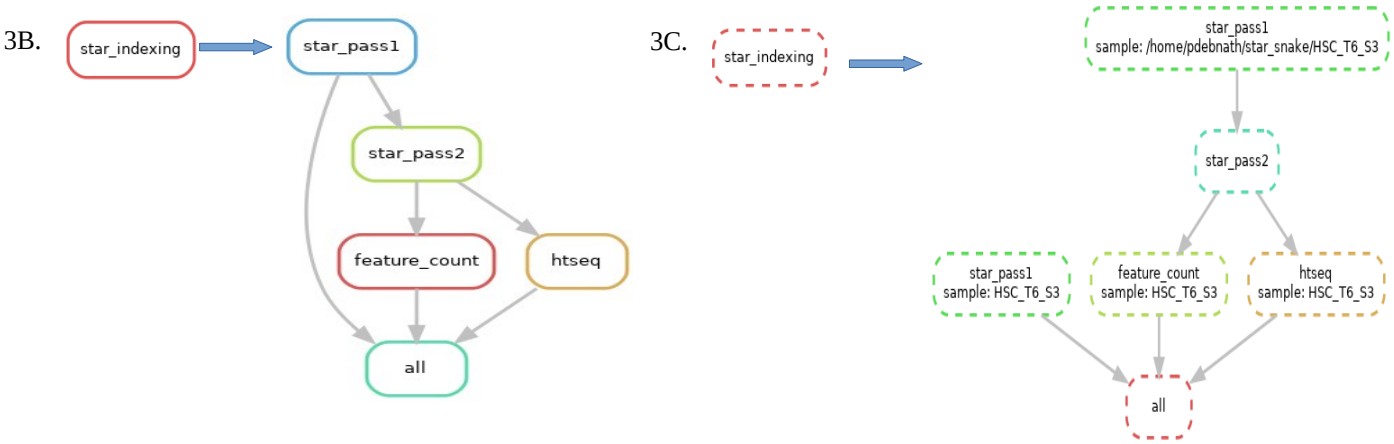


Fig. 3 Representation of Snakemake workflow of STAR indexing, STAR pass1, STAR pass2, FeatureCounts and HTSeq-count in filegraph (3A), rulegraph (3B) and DAG (3C).

HTSeq-qa The generated STAR aligned BAM file quality could be checked through python based *HTSeq-qa*. HTSeq-qa carried out in two steps. For HTSeq-qa the first step was to convert the aligned BAM file to SAM by the Samtools view (defined by rule converting_bam_to_sam_file in snakemake workflow). The next step was to perform quality assessment of the BAM file (describe by rule htseq_qa in snakemake workflow). For each rule, input file, output file and command lines were designated in shell block in snakefile. The overall HTSeq-qa workflow and output result are shown in the following figures:

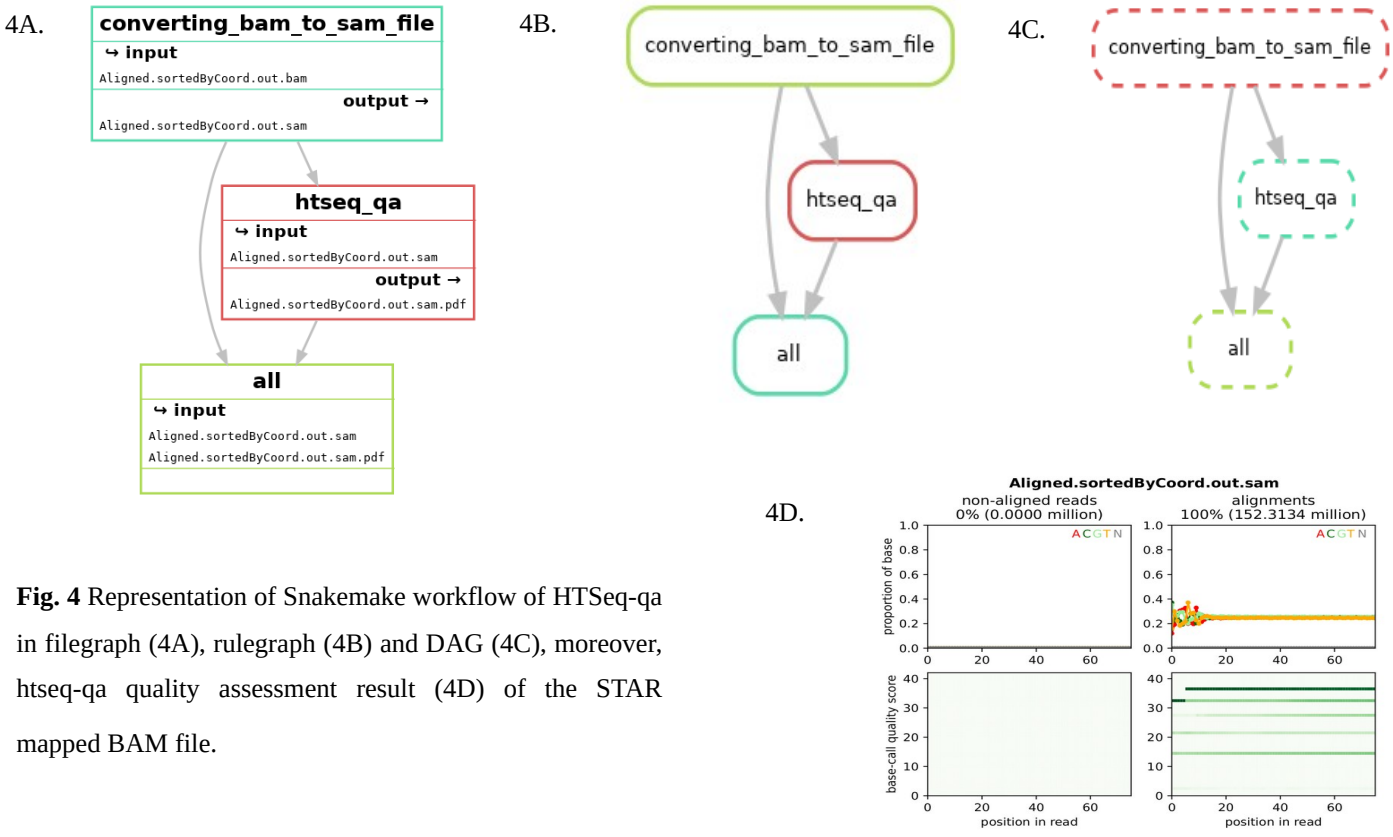


Fig. 4 Representation of Snakemake workflow of HTSeq-qa in filegraph (4A), rulegraph (4B) and DAG (4C), moreover, htseq-qa quality assessment result (4D) of the STAR mapped BAM file.

TPM counting Transcript per million is a normalization method for RNA-seq data. Transcripts Per Million ratio counted in the following ways (Li et al., 2010) (Pachter, 2011) :

1. Counting RPK (Reads per Kilobase): Each gene read count had divided by each gene length.

$$RPK = \text{Read Counts} / (\text{gene length} * 0.001)$$

2. Generating per million scaling factor and counting TPM: Summed up all of the RPK values and divided the number by 1000,000. And then the RPK values divided with per million scaling factor and TPM would be counted.

$$\text{Per million scaling factor} = \Sigma RPK / 10^6$$

$$TPM = \text{Each gene RPK values/per million scaling factor}$$

TPM counting was performed by the integration of Jupyter notebook to the Snakemake workflow by using the notebook block instead of shell block in Snakefile.

In the following chapter, we have discussed about the potentiality and validation of generated Snakemake workflow regarding analyzing paired-end RNA-seq data. Moreover, we have compared the alignment independent (Kallisto and Salmon) and alignment dependent tools (FeatureCounts and HTSeq-count) on the basis of Transcript Per Million (TPM) values and alike counted reads.

3. Result and Discussion

Effectiveness of generated Snakemake workflow pipeline Paired-end RNA-seq data of Norway rat (*Rattus norvegicus*) was used to generate the pipeline. We have implemented the same pipeline on the other paired-end RNA-seq data of Human (*Homo sapiens*) and the pipeline worked well. Therefore, for any types of paired-end sample, this workflow is automated and efficient for working with FastQC, MultiQC, Cutadapt, Kallisto, Salmon, STAR indexing and mapping, FeatureCounts, Htseq-count and Htseq-qa. Moreover, It's been observed that for sustainable data analysis, provided Snakemake workflow is worthwhile. The first effective thing is its readability and transparency, because the correlation of output to the input, the script and the code is undoubtedly understandable. Moreover, through the pipeline we can skip the reoccurrence of the samples. Furthermore, the order of the executed project is revealed through the defined collection of rules in snakefile, consequently unnecessary steps can be skipped effortlessly. For fruitful Snakemake workflow pipeline, some points should be considered. The placeholders, rule names or declarations, and wildcards should be valid. Avoidance of mismatched wildcards, write-protected output files, functional problems, unbalanced quotes or brackets, wrong indentation and target output files must be determined accurately. Finally, the commas or colons must be checked properly.

Comparison between alignment independent and alignment dependent tools We had compared the alignment independent tools (Kallisto and Salmon) and alignment dependent tools (FeatureCounts and Htseq-count) according to the Transcript per million count (TPM) result and mutual counted reads. The comparison was performed on our two selected organisms Norway rat (*Rattus norvegicus*) and Human (*Homo sapiens*).

In the following table we have compared the *top 6* Transcripts Per Million (TPM) values counted by alignment independent tools (Kallisto and Salmon) and alignment dependent tools (FeatureCounts) in both *Rattus norvegicus* and *Homo sapiens*. Where, in Kallisto and Salmon TPM value is higher than the FeatureCounts.

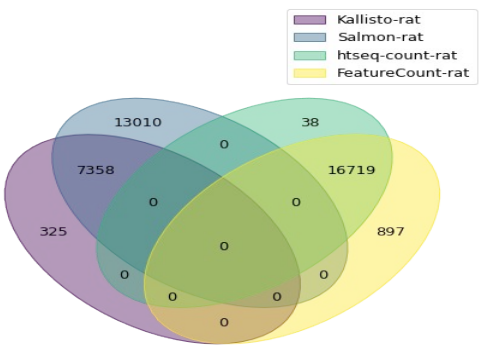
Table 1 Comparison of the Transcripts Per Million (TPM) values counted by Kallisto, Salmon and FeatureCounts in *Rattus norvegicus* and *Homo sapiens.*

Rattus norvegicus						Homo sapiens					
Kallisto		Salmon		FeatureCounts		Kallisto		Salmon		FeatureCounts	
Target ID.	TPM	Target ID.	TPM	Gene ID.	TPM	Target ID.	TPM	Target ID.	TPM	Gene ID.	TPM
ENSRNOT00000030919.5	9138.88	ENSRNOT00000030919.5	10373.82	ENSRNOG00000034254	7866.68	ENST00000621981.1	133658	ENST00000637028.1	196241.13	ENSG00000034510	10415.69
ENSRNOT00000022846.5	8660.06	ENSRNOT00000022846.5	7527.03	ENSRNOG00000010208	6765.1	ENST00000580862.1	51281	ENST00000579335.1	39734.02	ENSG00000087086	8305.23
ENSRNOT00000005577.7	7311.26	ENSRNOT00000013608.6	6316.14	ENSRNOG00000000957	6047.25	ENST00000627191.1	45502.9	ENST00000638396.1	35381.53	ENSG00000184009	8272.82
ENSRNOT00000013393.6	6780.41	ENSRNOT00000001265.5	5625.82	ENSRNOG00000009439	5742.49	ENST00000639535.1	34187.3	ENST00000581765.1	27420.69	ENSG00000075624	6637.83
ENSRNOT00000013745.7	6103.07	ENSRNOT00000018820.5	5464.92	ENSRNOG00000001148	5624.32	ENST00000619176.1	32565.6	ENST00000561440.2	13642.05	ENSG00000198840	6636.22
ENSRNOT00000013608.6	5636.08	ENSRNOT00000013745.7	5375.76	ENSRNOG00000018630	5080.89	ENST00000622012.1	26990	ENST00000387347.2	13438.85	ENSG00000198712	3615.62

Transcripts per million (TPM) value is higher in alignment independent tools (Kallisto and Salmon) because they are involved in transcript-level quantification rather than gene-level quantification. In consequence, the TPM result is more accurate in Kallisto and Salmon, instead of FeatureCounts.

Furthermore, for the comparison, we had compared the reads counted by Kallisto, Salmon, FeatureCounts and HTSeq-count. We tried to find out the similarity of counted reads within the alignment independent and alignment dependent tools. In case of Kallisto for Norway rat and Human total 7683 and 617 reads were counted, respectively. Whereas, by Salmon, 20368 and 17319 reads were counted, respectively in Norway rat and Human. Among the total counted reads for Norway rat and Human, Kallisto and Salmon both counted 7358 and 495 similar reads, respectively. Moreover, regarding HTSeq-count for Norway rat and Human total 16757 and 27115 reads were counted, respectively. While, by FeatureCounts, 17616 and 28492 reads were counted, respectively in Norway rat and Human. Among the total counted reads for Norway rat and Human, HTSeq-count and FeatureCounts both counted 16719 and 25959 similar reads, respectively. The overall correlation of homogeneous counted reads is represented in the Venn diagram.

5A.



5B.

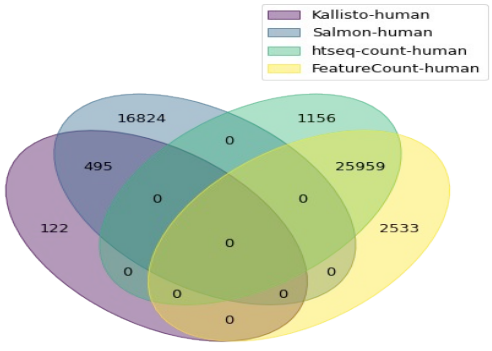


Fig. 5 Venn diagram representation of similar reads correlation between alignment independent tools such as Kallisto and Salmon and alignment dependent tools such as HTSeq-count and FeatureCounts in *Rattus norvegicus* (5A) and *Homo sapiens* (5B).

From the Venn diagram, we found that there are no similar counted reads for alignment independent and alignment dependent tools. There is no correlation within aligned independent tools (Kallisto and Salmon) and aligned depended tools (FeatureCounts and HTSeq-count), for the reason that, FeatureCounts and HTSeq-count deal with gene-level quatification rather than transcript-level quantification like Kallisto and Salmon.

5. Conclusion

It can be concluded that the Snakemake workflow is automated and reproducible enough for the analysis and re-analysis of paired-end RNA-seq dataset. Through the configuration file the workflow can easily customize. It is expected that the advances in automating the generated pipeline will facilitated to study the distinct type of RNA-seq data.

4. References

- Anders, S., Pyl, P. T., & Huber, W. (20). February 2014. HTSeq-a Python framework to work with high-throughput sequencing data. Bioinformatics http://dx. doi. org/10.1093/bioinformatics/btu638.
- Andrews, S. (2010). FastQC: a quality control tool for high throughput sequence data.
- Batut, B., Freeberg, M., Heydarian, M., Erxleben, A., Videm, P., Blank, C., ... & Delisle, L. (1970). Reference-based RNA-Seq data analysis.
- Bray, N. L., Pimentel, H., Melsted, P., & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. Nature biotechnology, 34(5), 525-527.
- Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. Nature biotechnology, 35(4), 316-319.
- Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., ... & Gingeras, T. R. (2013). STAR: ultrafast universal RNA-seq aligner. Bioinformatics, 29(1), 15-21.
- Ellson, J., Gansner, E., Koutsofios, L., North, S. C., & Woodhull, G. (2001, September). Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing* (pp. 483-484). Springer, Berlin, Heidelberg.
- Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics, 32(19), 3047-3048.
- Geraci, F., Saha, I., & Bianchini, M. (2020). RNA-Seq analysis: methods, applications and challenges. Frontiers in Genetics, 11, 220.
- Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. Bioinformatics, 28(19), 2520-2522.
- Larsonneur, E., Mercier, J., Wiart, N., Le Floch, E., Delhomme, O., & Meyer, V. (2018, December). Evaluating workflow management systems: A bioinformatics use case. In 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2773-2775). IEEE.
- Li, B., Ruotti, V., Stewart, R. M., Thomson, J. A., & Dewey, C. N. (2010). RNA-Seq gene expression estimation with read mapping uncertainty. Bioinformatics, 26(4), 493-500.
- Liao, Y., Smyth, G. K., & Shi, W. (2014). featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics, 30(7), 923-930.
- Mackenzie, R. J. (2018). RNA-seq: Basics, Applications and Protocol. Retrieved from Technology Networks: https://www. technologynetworks. com/genomics/articles/rna-seqbasics-applications-and-protocol-299461.
- Magoč, T., & Salzberg, S. L. (2011). FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics, 27(21), 2957-2963.

- Mölder, F., Jablonski, K. P., Letcher, B., Hall, M. B., Tomkins-Tinch, C. H., Sochat, V., ... & Köster, J. (2021). Sustainable data analysis with Snakemake. F1000Research, 10.
- Ozsolak F, & Milos PM. RNA sequencing: advances, challenges and opportunities. *Nat. Rev. Genet*,2011; 12(2), 87–98. doi:10.138/nrg2934
- Pachter, L. (2011). Models for transcript quantification from RNA-Seq. arXiv preprint arXiv:1104.3889.
- Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., & Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. Nature methods, 14(4), 417-419.