A workflow reproducibility scale for automatic validation of biological interpretation results

Hirotaka Suetake[1], **Tsukasa Fukusato**[2], **Takeo Igarashi**[1], **and Tazro Ohta**[3,✉]

[1]Department of Creative Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
[2]Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
[3]Database Center for Life Science, Joint Support-Center for Data Science Research, Research Organization of Information and Systems, Shizuoka, Japan

Background: **Reproducibility of data analysis workflow is a key issue in the field of bioinformatics. Recent computing technologies, such as virtualization, have made it possible to reproduce workflow execution with ease. However, the reproducibility of results is not well discussed; that is, there is no standard way to verify whether the biological interpretation of reproduced results are the same. Therefore, it still remains a challenge to automatically evaluate the reproducibility of results.**

Results: **We propose a new metric, a reproducibility scale of workflow execution results, to evaluate the reproducibility of results. This metric is based on the idea of evaluating the reproducibility of results using biological feature values (e.g., number of reads, mapping rate, and variant frequency) representing their biological interpretation. We also implemented a prototype system that automatically evaluates the reproducibility of results using the proposed metric. To demonstrate our approach, we conducted an experiment using workflows used by researchers in real research projects and the use cases that are frequently encountered in the field of bioinformatics.**

Conclusions: **Our approach enables automatic evaluation of the reproducibility of results using a fine-grained scale. By introducing our approach, it is possible to evolve from a binary view of whether the results are superficially identical or not to a more graduated view. We believe that our approach will contribute to more informed discussion on reproducibility in bioinformatics.**

Workflow | Provenance | Reproducibility
Correspondence: ***t.ohta@dbcls.rois.ac.jp***

DRAFT

Background

Bioinformatics is big data science and is considered the most demanding domain in terms of data acquisition, storage, distribution, and analysis (1). Because the low cost and high throughput of measurement instruments have made it possible to generate large amounts of data, large-scale data analysis using a computer is required to extract valuable knowledge from the data (2, 3). For each data analysis process, such as data transformation, public database referencing and merging, and statistical processing, much open-source software is developed and released (4). Researchers typically choose appropriate software for each analysis process, build a workflow by combining the software, and execute the workflow in a computing environment (5). However, it can be challenging to ensure the reproducibility of data analysis due to a number of factors, such as a large amount of data, the diversity of data types and software, and the complexity of the computing environment (6).

Reproducibility of research is an essential issue in the scientific community (7, 8). However, Baker raised the alarm of a "reproducibility crisis" based on survey results that *"more than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments"* (9, p. 452). The key here is the requirement for research to be considered reproducible. Drummond argued that replicability and reproducibility are often confused, but they are different concepts and need to be clearly distinguished (10). The Association for Computing Machinery (ACM) also attempts to define the terms repeatability, reproducibility, and replicability (Table 1) (11). While these definitions are in the context of computerized analysis, it should be noted that most existing studies have focused on whether the execution can be reproduced or not and have not considered the verification of the results. That is, they only state that the resulting data are exactly the same as in the original but do not adequately discuss the verification of whether the results are reproducible or not. Therefore, the reproducibility of data analysis can be divided into two parts: the execution of the analysis and the verification of the results. We will focus our discussion on the second part, verification.

Many workflow systems have been developed to improve the efficiency of building and executing complex data analysis (12–14). Each system has unique characteristics, but in particular, workflow systems can have a syntax for describing the data analysis, called a workflow language. Large user communities have been formed around these workflow languages. The Common Workflow Language (CWL) (15), the Workflow Description Language (WDL) (16), Nextflow (17), and Snakemake (18) are typical examples. These systems also have execution systems that work with computational frameworks, such as job schedulers, container runtimes, and package managers. Thus, these workflow systems facilitate the execution of data analysis by different teams and in different environments through the use of virtualization technology and syntax that abstracts software and computational requirements (19).

The advent and widespread use of workflow systems have facilitated data analysis re-execution. However, as mentioned earlier, to ensure reproducibility, it is necessary to verify the execution results, that is, whether the same biological interpretation is obtained or not. To address this issue, frameworks, such as Research Object Crate (RO-Crate) (20) and

Table 1. Repeatability, reproducibility, and replicability. According to the ACM (11), repeatability is defined as a researcher can reliably repeat her own computation. Reproducibility is defined as an independent group can obtain the same result using the author's own artifacts, and replicability is defined as an independent group can obtain the same result using artifacts which they develop completely independently.

Term	Team	Environment	Setup (code and data)	Result
Repeatability	Same	Same	Same	Same
Reproducibility	Different	Different	Same	Same
Replicability	Different	Different	Different	Same

CWLProv (21), have been proposed to generate workflow provenance, a structured archive that packages workflow-related metadata, such as workflow descriptions, execution parameters, input and output data, tests, and documentation, in a machine-readable format. This provenance information is distributed on workflow sharing platforms, such as WorkflowHub (22), Dockstore (23), and nf-core (24). When appropriate provenance is provided by the author, a researcher can use this information to verify new execution results making the process reproducible.

However, the process of comparing the provenance and execution results is often incomplete and inefficient. In automatic comparison, the checksums of the output files are used; however, they do not always match. This is because these checksums may differ depending on the software version, timestamps, heuristic algorithms, and computing environments (e.g., OS and CPU architecture, etc.). However, the same biological interpretation may be obtained even when the output files do not match exactly; for example, only the timestamps in the output files may differ. Thus, a simple comparison using a checksum is incomplete in verifying results. Another method is to have humans semantically interpret the results. However, due to its inefficiency, this method is not possible when the scale of the data analysis execution is large. From the above, the verification of results using provenance remains challenging because the current procedure is limited to incomplete automatic comparison and inefficient manual checking.

Automation is essential for the verification of practical workflows that output many files; however, binary determination by checking checksums is not sufficient. Thus, it is necessary to introduce a fine-grained scale to determine the degree of reproducibility of the results. Automatic verification of results using this scale will make verification of workflow reproducibility practical. In this paper, we propose a reproducibility scale of workflow execution results based on some discussion and experiments and a validation method using this scale. We implement a workflow execution system that generates a workflow provenance that contains metadata required for verification. This implementation is an extension of Sapporo (25), an existing workflow execution service (WES). Sapporo's extensibility makes it compatible with various workflow languages and execution systems. In addition, we adapt RO-Crate as the workflow provenance format. We also develop Tonkaz: a command-line tool that verifies the reproducibility of data analysis results by comparing the workflow provenances. To demonstrate the effectiveness of our approach, we apply it to workflows used by researchers in real research projects. The full reproducibility of research is still an issue that has not been fully resolved. Nevertheless, we hope our approach will contribute to solving this problem by increasing the resolution of the definition of reproducibility.

Methods

Reproducibility scale of workflow execution results. A workflow is a sequence of computational steps that combine analysis tools according to their inputs and outputs. The first tool takes input data and passes its output on as input for the next tool. Thus, the result of the workflow execution is the cumulative output of each tool or the last tool in the workflow. It should be noted that the output of a tool includes not only output files, but also execution logs (e.g., standard output and error) and runtime information (e.g., exit code, start time, and end time). Returning to the purpose of data analysis here, it is to obtain useful biological knowledge from the data. Therefore, it is not sufficient to consider the output files and logs as the only result of the workflow execution; the biological feature values interpreted from the output files and logs should be considered as the result of the workflow execution.

The format to represent biological features obtained from data analysis is not standardized and varies depending on the analysis tool. For example, there are summarized formats (tabular and graph) and formats that express biological features themselves, such as Sequence Alignment/Map (SAM) and the Variant Call Format (VCF). To interpret and verify the results, the individual executing the workflow visually checks the output graph or uses a tool to extract a numerical feature value from the file, for example, SAMtools (26) to extract mapping statistics from the SAM format. Because these processes require domain knowledge, it is ideal that the workflow itself provides a structured summary and a way to interpret it. However, this depends on the skill and effort of the individual workflow developer, and the diversity of tools and workflows makes it challenging to provide them in a standardized way.

There are several workflows that provide a way to verify reproducibility using biological feature values. For example, the RNA-seq workflow (27) distributed by the nf-core project has a test mode to verify that the workflow is working as expected. In this mode, the workflow is executed with a small test dataset, and the biological feature values are compared with the expected values. The mapping rate, which represents the percentage of reads that are mapped to the reference genome, is used as a biological feature value. If the difference

DRAFT

between the values is within the threshold, the workflow is considered to be working as expected. As a preliminary experiment, we compared the output files without using such biological feature values—that is, the checksum method was used to verify an exact match of a file. As a result, when we executed the above RNA-seq workflow twice in the same environment and compared the file output BAM files (the compressed binary version of the SAM files), we found that the checksum values were different and the file sizes differed by several bytes (see the Section "Results" for details). It is ideal that the output files are exactly the same, but it is difficult to achieve this goal because the output files are generated by the analysis tools, and these tools are not designed to produce the same output all the time. Therefore, we concluded that it is not sufficient to check only the exact match of output files to verify the reproducibility of workflow execution results and that a method using biological feature values and threshold should also be introduced.

Based on the above discussion and preliminary experiments, we propose a method to verify the reproducibility of workflow execution results using biological feature values and threshold. The method consists of two steps: (1) extracting biological feature values from the output files and logs and (2) comparing the extracted biological feature values with the expected values using threshold values. A detailed description of each step is provided in the Sections "Generation of workflow provenance containing biological feature values" and "Automatic verification of reproducibility." We also propose a scale to evaluate the reproducibility of workflow execution results based on the method (Table 2). This allows the reproducibility of results to be expressed at a higher resolution than a binary measure of whether the results are the same or not.

Generation of workflow provenance containing biological feature values. To verify the reproducibility of workflow execution results using biological feature values, it is necessary to package the workflow execution results as the workflow provenance in a standardized format. Because there are many workflow languages and execution engines, we first abstracted the workflow execution itself. Thus, we extended Sapporo, an existing WES implementation. Sapporo has an API scheme that satisfies the Global Alliance for Genomics and Health (GA4GH) Workflow Execution Service (WES) standard (28), enabling the workflow execution and results acquisition in a standardized manner. In addition, due to its extensibility, it can execute workflows written in various languages, such as CWL, WDL, Nextflow, and Snakemake. Therefore, by extending Sapporo, workflow execution written in various languages can be handled in the same way.

When a workflow is executed in Sapporo, the files related to the execution are stored in the file system as workflow provenance. This provenance directory contains the workflow definition files, input files, intermediate files, output files, log files, execution parameters, runtime information, etc. Thus, we converted Sapporo's provenance into RO-Crate, a standardized format for packaging research objects expressed in JSON-LD. Because the RO-Crate use case included the packaging of workflow execution results, it was sufficient to map Sapporo's provenance to the ontology provided by RO-Crate. However, for verification, we defined some additional terms and properties[1]. For example, we defined the property "mappedRate" to represent the mapping rate of the output file, which is a biological feature value used for verification. In addition, RO-Crate is designed to rely on the local file system for file location resolution and checksum representation. However, we prioritized the portability of being able to carry the provenance in a single file, so we put all the information necessary for verification, such as checksums, biological feature values, and contents of files of small size, in the RO-Crate file.

Because the workflow output freely produces a large number of diverse files, it is impractical to extract biological feature values for all files. Thus, we used the file extension to determine the file type and used an appropriate tool to extract the biological feature values. We used the file types defined in the EDAM ontology (29), which are widely used to express biological interpretations (Table 3). For example, if the file type is SAM, we used SAMtools to extract the number of reads and the mapping rate, and if the file type is VCF, we used VCFtools (30) to extract the number of variants and the variant frequency. In addition, the number of lines in the file is also a biological feature value. For example, if the file type is FASTQ, four lines represent one sequence read. Figure 1 is an example of a file entity in RO-Crate, which contains the biological feature values, such as the statistics obtained from the file, file size, and the number of lines.

For provenance enrichment and sharing, we integrated Sapporo and Yevis (31). Yevis is a system that builds a workflow registry and also acts as a client for Sapporo. The workflow metadata file used in Yevis contains not only the information required to execute the workflow but also the information for workflow availability and traceability in workflow sharing, such as author, open-source license, and documentation link. Thus, by executing the workflow in Sapporo via Yevis, the availability and traceability of the generated provenance are improved. In addition, because Yevis' workflow sharing feature enables the attachment of generated provenance to shared workflows, the reproducibility of shared workflows can be verified by other users.

From the above, by executing the workflow with Sapporo and Yevis (Figure 2), a workflow provenance containing feature values representing a biological interpretation is generated as RO-Crate. This method also applies to workflows written in various languages and can address a wide range of use cases, such as workflow sharing. Therefore, by generating and sharing a provenance containing biological feature values, it is possible to verify the reproducibility of the workflow execution results in other users' environments.

Automatic verification of reproducibility. We developed Tonkaz to automatically verify the reproducibility of work-

[1] https://raw.githubusercontent.com/sapporo-wes/sapporo-service/main/sapporo/ro-terms.csv

DRAFT

Table 2. Reproducibility scale of workflow execution results. For each of the output files, this is determined by comparing the expected provenance with the provenance of the actual execution. If the file of the same name in the expected provenance and the actual execution are identical, the file is considered to be fully reproduced. If the file of the same name is not identical, it is determined whether its difference is acceptable or not using the feature and threshold values. If the difference is acceptable, the file is considered to be partially reproduced. If the file exists in the expected provenance but not in the actual execution, it is considered to be not reproduced.

Reproducibility scale	Level	Description
Fully Reproduced	3	Output files are identical with the same checksum.
Acceptable Differences	2	Output files are not identical, but their biological feature values (e.g., number of reads, mapping rate, and variant frequency) are similar (within a threshold).
Unacceptable Differences	1	Output files are not identical, and their biological feature values are not similar (beyond threshold).
Not Reproduced	0	The workflow does not produce output files.

Table 3. File types and extensions defined in EDAM ontology. These file types and extensions are used to extract biological feature values from the output files.

EDAM ID	File type	Extension
`format_1929`	FASTA	`.fa, .fasta`
`format_1930`	FASTQ	`.fq, .fastq, .fq.gz, .fastq.gz`
`format_1975`	GFF	`.gff, .gff3`
`format_2306`	GTF	`.gtf`
`format_2572`	BAM	`.bam`
`format_2573`	SAM	`.sam`
`format_3003`	Bed	`.bed`
`format_3004`	BigBed	`.bb`
`format_3005`	Wig	`.wig`
`format_3006`	BigWig	`.bw`
`format_3016`	VCF	`.vcf, .vcf.gz`


```
{
  "@context": ["https://w3id.org/ro/crate/1.1/context", { ... }],
  "@graph": [
    ...,
    {
      "@id": "outputs/star_salmon/RAP1_UNINDUCED_REP2.markdup.sorted.bam",
      "@type": ["File", "FormalParameter", "OutputFile"],
      "contentSize": 3279083,
      "dateModified": "2022-09-08T08:52:19.755363",
      "encodingFormat": "application/gzip",
      "format": {
        "@id": "http://edamontology.org/format_2572"
      },
      "gid": 1000,
      "mode": "-rw-r--r--",
      "sha512": "2d6c8436dd1da0e4e49f9bdfbf8d656d7740f7eae149bb2add417d6739c05aeb441aa84239041f9aac87688042c7b31bfca5c95d0dfaf742512f2e740a788979",
      "stats": {
        "@id": "#31d3ba80-21df-4ab7-93e7-558154d07161"
      },
      "uid": 1000,
      "url": "http://localhost:1122/runs/93f4d8bf-424d-4d5e-bc79-9482e1620be9/data/outputs/star_salmon/RAP1_UNINDUCED_REP2.markdup.sorted.bam"
    },
    {
      "@id": "#31d3ba80-21df-4ab7-93e7-558154d07161",
      "@type": ["FileStats"],
      "duplicateRate": 0.8027008887713803,
      "duplicateReads": 78936,
      "generatedBy": {
        "@id": "#samtools"
      },
      "mappedRate": 1.0,
      "mappedReads": 98338,
      "totalReads": 98338,
      "unmappedRate": 0.0,
      "unmappedReads": 0
    },
    ...
  ]
}
```


Fig. 1. An example of a file entity in RO-Crate. This is a part of the actual workflow execution results and uses the RNA-seq workflow distributed by the nf-core project. The output file in this example is a BAM file, and its biological feature values are the file size, number of mapped reads, and mapping rate. Thus, the file entity contains properties such as "contentSize," "stats:mappedReads," and "stats:mappedRate". These are defined as additional terms in Sapporo, and the values are extracted from the file using SAMtools.

flow execution results by comparing the biological feature values contained in the workflow provenance. One use case of Tonkaz is to compare the expected result, which is provided by a workflow developer, and the actual result, which is generated in the user's environment (Figure 2). That is, Tonkaz verifies that the results are the same according to the ACM's definition of reproducibility (Table 1). Another use case is ACM's definition of repeatability, which is to verify that the results are the same even if a workflow is executed multiple times in the same environment, and it will not be broken by updates to dependencies. Thus, these use cases indicate that we must verify the reproducibility of the results, regardless of the differences in execution methods and environments.

We designed Tonkaz to accept as arguments two RO-Crates, one containing the expected provenance and the other containing the actual provenance. Tonkaz compares the biological feature values of the output files in the two RO-Crates and calculates the reproducibility scale for each file. The files to be compared are those output files that have been assigned EDAM ontology, as described in Table 3. This is because the workflow output files often include log files that are not related to the biological interpretation and image files that are not mechanically comparable. For example, nf-core's RNA-seq workflow produces 872 files, but only 25 files are assigned EDAM ontology. In the process of comparing files and calculating the reproducibility scale, Tonkaz first checks whether the files are identical using a checksum (Figure 3). If the files are identical, the reproducibility scale is "Fully Reproduced." If the files are not identical, Tonkaz compares the biological feature values of the files using a threshold value to determine whether the differences are acceptable or not. The default threshold value used is 0.05, but this value can be changed according to the use case. This is because some workflows for medical applications or quality control of biological materials require a lower threshold value. The comparison result is finally summarized in a table, and the reproducibility scale of each file is also displayed in a table (Figure 4). However, Tonkaz does not score the reproducibility of the entire workflow. This is because, again, the purpose of comparison may differ depending on the use case, and it is not practical to automate the final decision. Thus, we implemented an option to generate structured data in Tonkaz. In addition, we believe that workflow developers should use this option and write conditions or scripts to determine the reproducibility for each use case.

Results

To demonstrate the effectiveness of our approach, we verified the reproducibility of workflow execution results by comparing the results of public workflows used by researchers in real research projects, not simple ones for testing. This verification was based on the following five practical use cases: (1) execution in the same environment, (2) execution in a different environment, (3) execution of different versions of the workflow, (4) execution with missing datasets, and (5) comparison using all output files. We used the following three workflows: (1) the mitochondrial short variant discovery workflow distributed a GATK best practice workflow (hereafter referred to as GATK workflow, language: WDL) (32), (2) RNA-Seq workflow distributed by nf-core (hereafter referred to as RNA-seq workflow, language: Nextflow) (27), and (3) GATK best practice-compatible germline short variant discovery workflow, which is used to process whole-genome sequencing data of the Japanese Genotype-phenotype Archive (hereafter referred to as JGA workflow, language: CWL) (33). We used the following two execution environments: (1) Ubuntu 20.04 LTS (CPU: Intel Xeon E5-2640 @ 2.50GHz, RAM: 24GB, Docker: 20.10.8) and (2) macOS 12.5.1 (CPU: Apple M1 Max, RAM: 64GB, Docker: 20.10.16). Table 4 shows the setting for each execution as a combination, and Table 5 summarizes the verification results based on the use cases. The methods and results of the workflow execution and verification are described in the online documentation "sapporo-wes/tonkaz -tests/README.md" (34) and are published on Zenodo (35).

The Comparisons C1, C3, and C5 present execution results in the same environment. Comparison C1 was performed using the GATK workflow, and the output file types were BAM and VCF. The reproducibility scale value was Level 2 (Acceptable Difference) for all files, with no differences in biological feature values expressing biological interpretation (e.g., mapping rate and variant frequency). The difference between these files was due to the fact that both the BAM and VCF files included the file paths of the original input file and timestamps in the header lines. Thus, when using the analysis tool GATK (36), it is challenging to fully reproduce the output files because of the behavior that the output files contain the file paths and timestamps. Comparison C3 was performed using the JGA workflow, and the output file types were VCF. This result also showed no differences in biological feature values, and the differences in file contents were due to the behavior of GATK. Comparison C5 was performed using the RNA-seq workflow, and the output file types were BAM, GTF, and BED. All GTF and BED files were Level 3 (Fully Reproduced), and all BAM files were Level 2 (Acceptable Difference). The difference between BAM files was due to the different order of the mapped reads in the BAM file. These BAM files were mapped by STAR (37), and sorted by SAMtools; however, differences occurred. These results show cases in which the output files were not identical, although the biological feature values were equal, due to the behavior of the analysis tool.

The Comparisons C2, C4, and C6 present the execution results in different environments. All of these comparison results were Level 0 (Not Reproduced) because all execution in the Mac environment either failed or never finished. All workflows used in this experiment were Docker containerized and were designed to be very reproducible in the execution context; however, runtime errors occurred due to the Arm processor architecture of the Mac environment. Thus, even a very well-considered workflow may not be reproducible in a different environment. In such cases, it is essential to increase the debuggability of the cause of the irrepro-

DRAFT

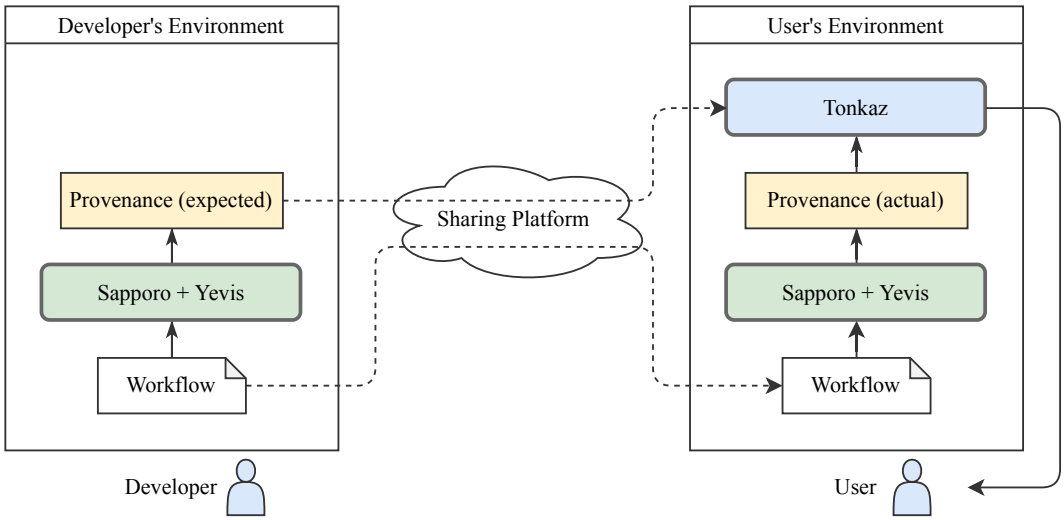

Fig. 2. The flowchart representing the Tonkaz use case. The workflow built by the workflow developer is executed by WES, which is a combination of Sapporo and Yevis, and the workflow provenance, including feature values of the output files, is generated in RO-Crate format. This provenance is used as the expected value for the verification of reproducibility. Using the shared workflow, the user executes the shared workflow in his/her own environment using WES. Using Tonkaz, the user then compares the shared provenance with the provenance generated by the user's workflow execution and verifies the reproducibility of the workflow execution results.

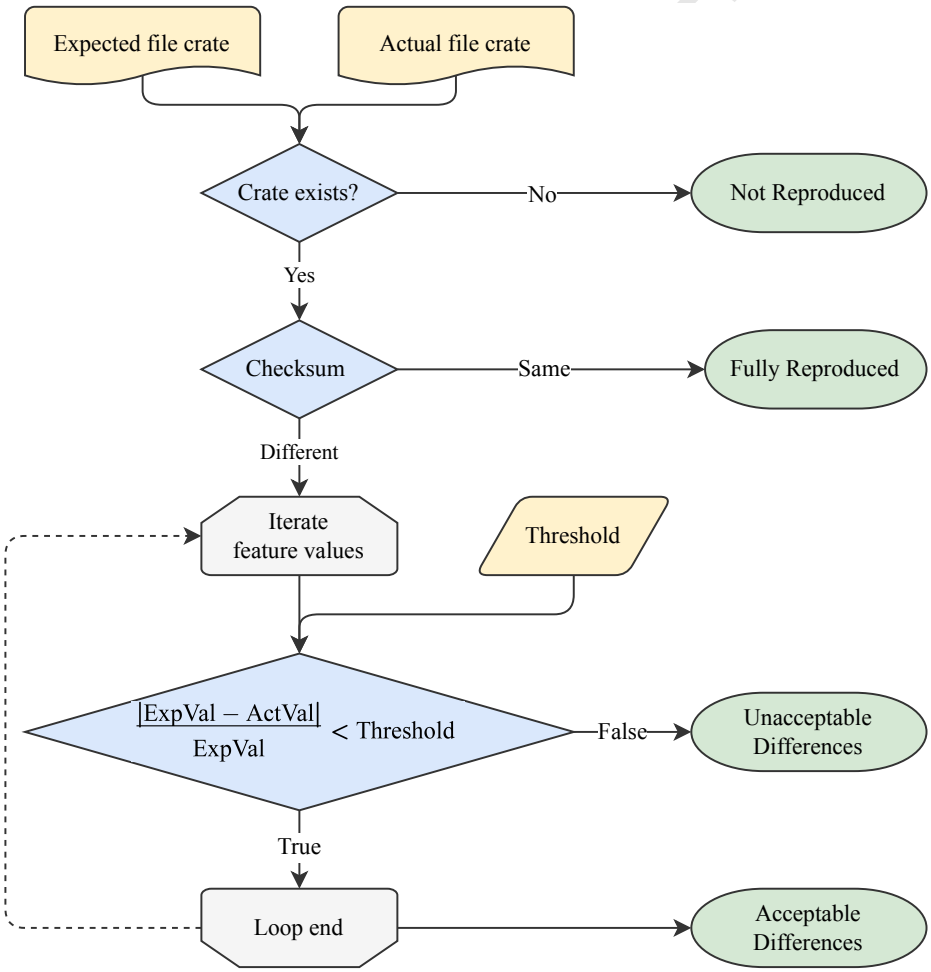

Fig. 3. The process for calculating the reproducibility scale of a file. Tonkaz first checks whether the files are identical using a checksum. If the files are identical, the reproducibility scale value is "Fully Reproduced." If the files are not identical, Tonkaz compares the biological feature values of the files using a threshold value to determine whether the differences are acceptable or not. If the differences are acceptable, the reproducibility scale value is "Acceptable Difference." If the differences are unacceptable, the reproducibility scale value is "Unacceptable Difference." The default threshold value used is 0.05, but this value can be changed according to the use case. If the file entity exists only in one of the two RO-Crates, the reproducibility scale value is "Not Reproduced."

```
...

=== Level3 ⭐⭐⭐ (Same Checksum, 13/25 files)

  - star_salmon/rseqc/junction_annotation/bed/RAP1_IAA_30M_REP1.junction.Interact.bed
  - star_salmon/rseqc/junction_annotation/bed/RAP1_IAA_30M_REP1.junction.bed
  - star_salmon/rseqc/junction_annotation/bed/RAP1_UNINDUCED_REP1.junction.Interact.bed
  - star_salmon/rseqc/junction_annotation/bed/RAP1_UNINDUCED_REP1.junction.bed
  - star_salmon/rseqc/junction_annotation/bed/RAP1_UNINDUCED_REP2.junction.Interact.bed
  - star_salmon/rseqc/junction_annotation/bed/RAP1_UNINDUCED_REP2.junction.bed
  - star_salmon/rseqc/junction_annotation/bed/WT_REP2.junction.Interact.bed
  - star_salmon/rseqc/junction_annotation/bed/WT_REP2.junction.bed
  - star_salmon/stringtie/RAP1_IAA_30M_REP1.coverage.gtf
  - star_salmon/stringtie/RAP1_UNINDUCED_REP1.coverage.gtf
  - star_salmon/stringtie/RAP1_UNINDUCED_REP2.coverage.gtf
  - star_salmon/stringtie/WT_REP1.coverage.gtf
  - star_salmon/stringtie/WT_REP2.coverage.gtf

=== Level2 ⭐⭐ (Similar Features, 12/25 files)

  - star_salmon/WT_REP1.markdup.sorted.bam
    .-----------------------------------------------------------------.
    |                |        in Crate1        |        in Crate2        |
    |----------------|-------------------------|-------------------------|
    | File Size      | 6.86 MB (7196616)       | 6.86 MB (7192272)       |
    | Total Reads    | 188243                  | 188241                  |
    |   # Mapped     | 188243 (100.00%)        | 188241 (100.00%)        |
    |   # Duplicate  | 38470 (20.44%)          | 38470 (20.44%)          |
    '-----------------------------------------------------------------'

  - star_salmon/rseqc/junction_annotation/bed/WT_REP1.junction.Interact.bed
    .-----------------------------------------------------------------.
    |                |        in Crate1        |        in Crate2        |
    |----------------|-------------------------|-------------------------|
    | File Size      | 22.46 KB (22995)        | 22.32 KB (22856)        |
    | Line Count     | 162                     | 161                     |
    '-----------------------------------------------------------------'

  - star_salmon/stringtie/WT_REP2.transcripts.gtf
    .-----------------------------------------------------------------.
    |                |        in Crate1        |        in Crate2        |
    |----------------|-------------------------|-------------------------|
    | File Size      | 37.82 KB (38729)        | 37.82 KB (38729)        |
    | Line Count     | 259                     | 259                     |
    '-----------------------------------------------------------------'

...

=== Level1 ⭐ (Different Features, 0/25 files)

=== Level0 (Not Found, Crate1: 0 files, Crate2: 0 files)

Summarize compare result:

  .-----------------------------------------------------------------------------.
  |      Reproducibility      |   Level   |     Definition      |   File #    |
  |---------------------------|-----------|---------------------|-------------|
  | Fully Reproduced          | ⭐⭐⭐    | Same Checksum       | 13 files    |
  | Acceptable Differences    | ⭐⭐      | Similar Features    | 12 files    |
  | Unacceptable Differences  | ⭐        | Different Features  | 0 files     |
  | Not Reproduced            |           | Not Found           | 0 files     |
  '-----------------------------------------------------------------------------'
```

Fig. 4. Example of the Tonkaz output. Tonkaz displays a table for each file and a final summary table. The user checks those summary tables to determine the reproducibility of the entire workflow and the differences between the expected and actual files (e.g., by using the diff command).

Table 4. Combination table of workflow execution and execution settings. The first column is the definition of the execution name. In the second column and below are the workflow execution settings. The blank cells in the third and fourth columns indicate that there are no differences in the execution settings.

Execution name	Workflow	Version	Dataset	Environment
GATK_1st				Linux
GATK_2nd	GATK			Linux
GATK_mac				Mac
JGA_1st				Linux
JGA_2nd	JGA			Linux
JGA_mac				Mac
RNA-seq_1st		v3.7	Standard	Linux
RNA-seq_2nd		v3.7	Standard	Linux
RNA-seq_mac	RNA-seq	v3.7	Standard	Mac
RNA-seq_v3.6		v3.6	Standard	Linux
RNA-seq_small		v3.7	Small	Linux

Table 5. Comparisons of execution and verification results. The definition of each execution is defined in Table 4. Five use cases are assigned according to the combination of executions. In the fifth column and below are the number of files for each reproducibility scale defined in Table 2: Level 3 is "Fully Reproduced," Level 2 is "Acceptable Difference," Level 1 is "Unacceptable Difference," and Level 0 is "Not Reproduced."

ID	Source execution	Target execution	Use case	Level 3	Level 2	Level 1	Level 0
C1	GATK_1st	GATK_2nd	Same environment	0	5	0	0
C2	GATK_1st	GATK_mac	Different environment	0	0	0	5
C3	JGA_1st	JGA_2nd	Same environment	0	4	0	0
C4	JGA_1st	JGA_mac	Different environment	0	0	0	4
C5	RNA-seq_1st	RNA-seq_2nd	Same environment	20	5	0	0
C6	RNA-seq_1st	RNA-seq_mac	Different environment	0	0	0	25
C7	RNA-seq_1st	RNA-seq_v3.6	Different version	13	12	0	0
C8	RNA-seq_1st	RNA-seq_small	Missing dataset	8	5	7	5
C9	RNA-seq_1st	RNA-seq_2nd	All output files	557	306	1	8

DRAFT

ducibility of the execution results. Therefore, the importance of this debuggability indicates that it is helpful to include information about the execution environment in the workflow provenance; our approach and RO-Crate address them.

Comparison C7 presents the execution results in different versions. Workflow developers often check for workflow breakage when updating versions of analysis tools included in the workflow. In the RNA-seq workflow used in this comparison, the dependent analysis tools STAR, SAMtools, and StringTie (38) were updated with the workflow update from v3.6 to v3.7. As a result of the comparison (C7), the number of files with Level 2 increased compared to C5, a comparison involving the same version. The file types that became Level 2 were GTF, BED, and BAM; the GTF and BED files were newly changed from Level 3 to Level 2 when compared to C5. The differences between the GTF files were due to differences in the FPKM field values and the version of StringTie included in the header line. The BED files had a different number of lines, and the BAM files had a different number of mapped reads; however, those differences were within the threshold value. This result indicates that verification using biological feature values and threshold is effective because apparent differences occur in output due to version updates and other reasons, and it is necessary to determine whether these differences are acceptable or not.

Comparison C8 presents the execution results in a case where the input dataset was partially missing. The dataset used in RNA-seq_1st contains six sequence read files (FASTQ) (39), while the dataset used in RNA-seq_small contained four sequence read files (40). As a result of the comparison, the output files related to the sample with half the number of reads were Level 1 (Unacceptable Difference), while the sample with zero reads was Level 0. In this case, setting the threshold used for verification to, for example, 0.5 instead of 0.05 (default value) will verify that the workflow is functioning as expected. That is, this suggests that the threshold value and final decision may vary depending on the objectives of developers and users.

Comparison C9 presents the execution results in a case where all the output files were compared. Most of the files were Level 3 or Level 2; however, 16 files were not reproduced (Level 0). These level 0 files had random names or timestamps in the file names, for example, `mqc_mqc_mplplot_gtnuqiebfc_1.pdf` and `execution_report_2022-09-08_06-28-19.html`. Therefore, it is not appropriate to use all files to verify the reproducibility of execution results; it is essential to focus on characteristic files, such as BAM and GTF files.

For the five practical use cases, we found that our approach was well suited to verify the reproducibility of the workflow execution results. In all use cases, existing methods that use checksums to verify exact file matches can produce false pos-

itives; this means that the workflow is considered not reproduced, even though it is working as expected. Therefore, it is important to introduce a reproducibility scale and verify the workflow execution results' reproducibility at higher resolutions.

Discussion

Despite its complexity, data analysis in bioinformatics is considered reproducible and is being shared. In particular, the workflows shared by nf-core and GATK best practices are well maintained and include test datasets, documentation, and open-source licenses. Ideally, all shared workflows would be like these; however, in reality, this is challenging because of the amount of work and domain knowledge required. Thus, we aim to facilitate workflow sharing by providing a workflow provenance model and a workflow provenance verification method. However, our approach is not applicable in domains where it is difficult to verify the results and inferences using a computer. In such cases, it is first necessary to discuss an ontology or structured format for representing the research.

A related project, CODECHECK (41), aims to provide the verification of the reproducibility of data analysis by a third party in scientific publishing. CODECHECK proposes a procedure similar to a peer review system, in which the workflow associated with research articles is verified at the time of publication by a reviewer called a CODECHECKER. However, this project focuses on increasing the availability of the workflow, and does not verify the execution results. As such, it is unlikely to address the case of our concern that the execution results are not exactly the same, but the conclusions of the study remain the same. Our proposed metrics, a reproducibility scale of workflow execution results, would be useful in such workflow reproducibility validation in publishing as well.

Software begins to degrade from the moment it is developed, and it is not easy to maintain the same quality over time. Cases in which an error, including a stack trace, is thrown are quite fortunate; in many cases, the software cannot be executed in the first place, the process does not finish, or the output is inaccurate without throwing an error. Dealing with such cases and improving debuggability is accomplished by packaging the expected behavior of the software at the time it is developed. In our approach, we were able to attach information, such as OS, CPU architecture, and dependent software versions, to the expected workflow provenance due to RO-Crate's extensibility. However, when analysis tools are used internally, as in a workflow, the behavior of the analysis tools tends to be a black box. Therefore, if an option to display the reproducibility of the execution for each analysis tool is provided, it will be possible to identify the cause of the irreproducibility of the workflow execution results.

We used the file extension to determine the file type and specific tools to extract the biological interpretation from the workflow output. However, using the file extension is not always reliable, and extracting the biological interpretation from the file contents is not always possible. Thus, we also tried to take advantage of the summaries and logs generated during the execution of the analysis tools; however, we abandoned this approach due to the diversity and unstructured nature of the summaries and logs. MultiQC is an existing project that attempts to summarize the results of multiple analysis tools (42). We believe that integrating our approach with MultiQC will allow us to extract richer information from the workflow output and improve verification accuracy.

In the article regarding a "reproducibility crisis," Baker quoted a Johns Hopkins microbiologist as stating, *"The next step may be identifying what is the problem and to get a consensus."* (9, p. 452). Subsequently, the proliferation of virtualization technology and workflow systems has lowered the bar for re-executing data analysis that an individual or others have previously built. Despite this, workflow developers are always anxious about whether their workflows are broken. In response to this anxiety, we realized that the cause is our binary view of whether the workflow could be reproduced or not. To remove this anxiety, we proposed a new approach to verify the reproducibility of workflows by providing a range of reproducibility of execution results. With the development of sharing platforms, workflow sharing has become more active. Therefore, we hope that by verifying reproducibility and sharing the results, more workflows will be reused with confidence, which, in turn, will lead to increased scientific progress.

Availability of source code and requirements

- Project name: Tonkaz
- Project home page: https://github.com/sapporo-wes/tonkaz
- DOI: 10.5281/zenodo.7102376
- biotoolsID: tonkaz
- Operating system(s): Platform independent
- Programming language: TypeScript
- Other requirements: Deno
- License: Apache License, Version 2.0

- Project name: Sapporo-service
- Project home page: https://github.com/sapporo-wes/sapporo-service
- DOI: 10.5281/zenodo.7088999
- biotoolsID: sapporo-service
- Operating system(s): Platform independent
- Programming language: Python
- Other requirements: Docker recommended
- License: Apache License, Version 2.0

- Project name: Yevis-cli

DRAFT

- Project home page: https://github.com/sapporo-wes/yevis-cli
- DOI: 10.5281/zenodo.7088957
- biotoolsID: yevis-cli
- Operating system(s): Platform independent
- Programming language: Rust
- Other requirements: Docker recommended
- License: Apache License, Version 2.0

Availability of supporting data and materials

The workflow, result, and documentation related to the experiment described in the Section "Results" are available on GitHub and Zenodo as follows:

- Execution method and result (34)
- Workflow definitions (43)
- Raw data of workflow execution results (35)

Declarations

List of abbreviations. ACM: Association for Computing Machinery; CPU: Central Processing Unit; CWL: Common Workflow Language; GA4GH: Global Alliance for Genomics and Health; ID: Identifier; OS: Operating System; RO-Crate: Research Object Crate; SAM: Sequence Alignment/Map; VCF: Variant Call Format; WDL: Workflow Description Language; WES: Workflow Execution Service;

Ethical Approval. Not applicable for this study.

Consent for publication. Not applicable for this study.

Competing Interests. The authors declare that they have no competing interests.

Funding. This study was supported by JSPS KAKENHI (Grant Number 20J22439, assigned to H.S.), the CREST program of the Japan Science and Technology Agency (Grant Number JPMJCR17A1, assigned to T.I.), and the Life Science Database Integration Project, NBDC of Japan Science and Technology Agency.

Author's Contributions. H.S. and T.O. conceived and developed the methodology and software and conducted the investigation. H.S., T.F., and T.O. wrote the manuscript. T.F., T.O., and T.I. supervised the project. All authors read and approved the final version of the manuscript.

Acknowledgements

We acknowledge and thank the following scientific communities and their collaborative events where several of the authors engaged in irreplaceable discussions and development throughout the project: the Pitagora Meetup, Workflow Meetup Japan, NBDC/DBCLS BioHackathon Series, and RO-Crate community. We also acknowledge Prof. Masahiro Kasahara for his valuable comments on the project.

References

1. Zachary D. Stephens, Skylar Y. Lee, Faraz Faghri, Roy H. Campbell, Chengxiang Zhai, Miles J. Efron, et al. Big data: Astronomical or genomical? *PLOS Biology*, 13(7):e1002195, 2015. doi: 10.1371/journal.pbio.1002195.
2. Lincoln D. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207, 2010. doi: 10.1186/gb-2010-11-5-207.
3. Sara Goodwin, John D. McPherson, and Richard W. McCombie. Coming of age: Ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016. doi: 10.1038/nrg.2016.49.
4. Pjotr Prins, Joep de Ligt, Artem Tarasov, Ritsert C. Jansen, Edwin Cuppen, and Philip E. Bourne. Toward effective software solutions for big biology. *Nature Biotechnology*, 33(7): 686–687, 2015. doi: 10.1038/nbt.3240.
5. Jeffrey M. Perkel. Workflow systems turn raw data into scientific knowledge. *Nature*, 573 (7772):149–151, 2019. doi: 10.1038/d41586-019-02619-z.
6. Anna Bánáti, Péter Kacsuk, and Miklós Kozlovszky. Evaluating the reproducibility cost of the scientific workflows. In *2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 187–190, 2016-05. doi: 10.1109/saci.2016.7507367.
7. Software with impact. *Nature Methods*, 11(3):211–211, 2014. doi: 10.1038/nmeth.2880.
8. Rebooting review. *Nature Biotechnology*, 33(4):319–319, 2015. doi: 10.1038/nbt.3202.
9. Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, 2016. doi: 10.1038/533452a.
10. Chris Drummond. Replicability is not reproducibility: Nor is it good science. In *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*, Volume 1, 2009.
11. Association for Computing Machinery. Artifact review and badging version 1.1, 2020. https://www.acm.org/publications/policies/artifact-review-and-badging-current.
12. Felipe da Veiga Leprevost, Valmir C. Barbosa, Eduardo L. Francisco, Yasset Perez-Riverol, and Paulo C. Carvalho. On best practices in the development of bioinformatics software. *Frontiers in Genetics*, 5, 2014. doi: 10.3389/fgene.2014.00199.
13. Laura Wratten, Andreas Wilm, and Jonathan Göke. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature Methods*, 18(10):1161–1168, 2021. doi: 10.1038/s41592-021-01254-9.
14. Peter Amstutz, Maxim Mikheev, Michael R. Crusoe, Nebojša Tijanić, and Samuel Lampa. Existing workflow systems, 2021. https://s.apache.org/existing-workflow-systems.
15. Michael R. Crusoe, Sanne Abeln, Alexandru Iosup, Peter Amstutz, John Chilton, Nebojša Tijanić, et al. Methods included: Standardizing computational reuse and portability with the common workflow language. *arXiv*, 2021. doi: 10.48550/arXiv.2105.07028.
16. Kate Voss, Jeff Gentry, and Geraldine Van Der Auwera. Full-stack genomics pipelining with GATK4 + WDL + Cromwell. *F1000Research*, 2017. doi: 10.7490/f1000research.1114631.1.
17. Paolo Di Tommaso, Maria Chatzou, Evan W. Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319, 2017. doi: 10.1038/nbt.3820.
18. Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012. doi: 10.1093/bioinformatics/bts480.
19. Felipe da Veiga Leprevost, Björn A. Grüning, Saulo Alves Aflitos, Hannes L. Röst, Julian Uszkoreit, Harald Barsnes, et al. BioContainers: An open-source and community-driven framework for software standardization. *Bioinformatics*, 33(16):2580–2582, 2017. doi: 10.1093/bioinformatics/btx192.
20. Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José M. Fernández, et al. Packaging research artefacts with RO-Crate. *Data Science*, 5(2): 97–138, 2022. doi: 10.3233/ds-210053.
21. Farah Zaib Khan, Stian Soiland-Reyes, Richard O. Sinnott, Andrew Lonie, Carole Goble, and Michael R. Crusoe. Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. *GigaScience*, 8(11):giz095, 2019. doi: 10.1093/gigascience/giz095.
22. Carole Goble, Stian Soiland-Reyes, Finn Bacall, Stuart Owen, Alan Williams, Ignacio Eguinoa, et al. Implementing FAIR digital objects in the EOSC-Life workflow collaboratory. *Zenodo*, 2021. doi: 10.5281/zenodo.4605654.
23. Brian D. O'Connor, Denis Yuen, Vincent Chung, Andrew G. Duncan, Xiang Kun Liu, Janice Patricia, et al. The Dockstore: Enabling modular, community-focused sharing of Docker-based genomics tools and workflows. *F1000Research*, 6:52, 2017. doi: 10.12688/f1000research.10137.1.
24. Philip A. Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm, et al. The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, 38(3):276–278, 2020. doi: 10.1038/s41587-020-0439-x.
25. Hirotaka Suetake, Tomoya Tanjo, Manabu Ishii, Bruno P. Kinoshita, Takeshi Fujino, Tsuyoshi Hachiya, et al. Sapporo: A workflow execution service that encourages the reuse of workflows in various languages in bioinformatics. *F1000Research*, 11:889, 2022. doi: 10.12688/f1000research.122924.1.
26. Petr Danecek, James K. Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O. Pollard, et al. Twelve years of SAMtools and BCFtools. *GigaScience*, 10(2):giab008, 2021. doi: 10.1093/gigascience/giab008.
27. Harshil Patel, Phil Ewels, Alexander Peltzer, Rickard Hammarén, Olga Botvinnik, Gregor Sturm, et al. nf-core/rnaseq: nf-core/rnaseq v3.7, 2022. doi: 10.5281/zenodo.6513815.
28. Heidi L. Rehm, Angela J.H. Page, Lindsay Smith, Jeremy B. Adams, Gil Alterovitz, Lawrence J. Babb, et al. GA4GH: International policies and standards for data sharing across genomic research and healthcare. *Cell Genomics*, 1(2):100029, 2021.
29. Jon Ison, Matúš Kalas, Inge Jonassen, Dan Bolser, Mahmut Uludag, Hamish McWilliam, et al. EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, 29(10):1325–1332, 2013. doi: 10.1093/bioinformatics/btt113.
30. Petr Danecek, Adam Auton, Goncolo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, et al. The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158,

DRAFT

2011. doi: 10.1093/bioinformatics/btr330.
31. Hirotaka Suetake, Tsukasa Fukusato, Takeo Igarashi, and Tazro Ohta. Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows. *bioRxiv*, 2022. doi: 10.1101/2022.07.08.499265.
32. Anders Peterson, Axel Verdier, Ayman Abdel Ghany, Ben Bimber, Byunggil Yoo, Daniel Gómez-Sánchez, et al. broadinstitute/gatk - scripts/mitochondria_m2_wdl, 2021. https://github.com/broadinstitute/gatk/tree/33bda5e08b6a09b40a729ee525d2e3083e0ecdf8/scripts/mitochondria_m2_wdl.
33. Yuichi Kodama, Jun Mashima, Takehide Kosuge, Toshiaki Katayama, Takatomo Fujisawa, Eli Kaminuma, et al. The DDBJ Japanese Genotype-phenotype Archive for genetic and phenotypic human data. *Nucleic Acids Research*, 43(D1):D18–D22, 2015.
34. Hirotaka Suetake. sapporo-wes/tonkaz - tests/README.md, 2022. https://github.com/sapporo-wes/tonkaz/blob/main/tests/README.md. doi: 10.5281/zenodo.7102376.
35. Hirotaka Suetake. Raw data of workflow execution results used in Tonkaz's experiments, 2022. doi: 10.5281/zenodo.7098337.
36. Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, et al. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20(9):1297–1303, 2010. doi: 10.1101/gr.107524.110.
37. Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, et al. STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013. doi: 10.1093/bioinformatics/bts635.
38. Mihaela Pertea, Geo M. Pertea, Corina M. Antonescu, Tsung-Cheng Chang, Joshua T. Mendell, and Steven L. Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature Biotechnology*, 33(3):290–295, 2015. doi: 10.1038/nbt.3122.
39. nf-core Community. nf-core/test-datasets - rnaseq/samplesheet/v3.4/samplesheet_test.csv, 2018. https://raw.githubusercontent.com/nf-core/test-datasets/rnaseq/samplesheet/v3.4/samplesheet_test.csv.
40. Hirotaka Suetake and Tazro Ohta. sapporo-wes/test-workflow: 1.0.2 - assets/nf-core_rnaseq_samplesheet_small_test.csv, 2022. https://raw.githubusercontent.com/sapporo-wes/test-workflow/1.0.2/assets/nf-core_rnaseq_samplesheet_small_test.csv. doi: 10.5281/zenodo.7102664.
41. Daniel Nüst and Stephen J. Eglen. CODECHECK: An Open Science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility. *F1000Research*, 10:253, 2021. doi: 10.12688/f1000research.51738.1.
42. Philip Ewels, Måns Magnusson, Sverker Lundin, and Max Käller. MultiQC: Summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, 32(19):3047–3048, 2016. doi: 10.1093/bioinformatics/btw354.
43. Hirotaka Suetake and Tazro Ohta. sapporo-wes/test-workflow: 1.0.2, 2022. doi: 10.5281/zenodo.7102664.

DRAFT