

GigaScience

Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows

--Manuscript Draft--

Manuscript Number:	GIGA-D-22-00187R1	
Full Title:	Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows	
Article Type:	Research	
Funding Information:	JSPS (20J22439)	Not applicable
	Japan Science and Technology Agency (JP-MJCR17A1)	Not applicable
Abstract:	<p>Background: Many open-source workflow systems have made bioinformatics data analysis procedures portable. Sharing these workflows provides researchers easy access to high-quality analysis methods without the requirement of computational expertise. However, published workflows are not always guaranteed to be reliably reusable. Therefore, a system is needed to lower the cost of sharing workflows in a reusable form. Results: We introduce Yevis, a system to build a workflow registry that automatically validates and tests workflows to be published. The validation and test are based on the requirements we defined for a workflow being reusable with confidence. Yevis runs on GitHub and Zenodo and allows workflow hosting without the need of dedicated computing resources. A Yevis registry accepts workflow registration via a GitHub pull request, followed by an automatic validation and test process for the submitted workflow. As a proof of concept, we built a registry using Yevis to host workflows from a community to demonstrate how a workflow can be shared while fulfilling the defined requirements. Conclusions: Yevis helps in the building of a workflow registry to share reusable workflows without requiring extensive human resources. By following Yevis's workflow-sharing procedure, one can operate a registry while satisfying the reusable workflow criteria. This system is particularly useful to individuals or communities that want to share workflows but lacks the specific technical expertise to build and maintain a workflow registry from scratch.</p>	
Corresponding Author:	Tazro Ohta JAPAN	
Corresponding Author Secondary Information:		
Corresponding Author's Institution:		
Corresponding Author's Secondary Institution:		
First Author:	Hiroataka Suetake	
First Author Secondary Information:		
Order of Authors:	Hiroataka Suetake Tsukasa Fukusato Takeo Igarashi Tazro Ohta	
Order of Authors Secondary Information:		
Response to Reviewers:	# Point-by-point response to reviewers' comments GIGA-D-22-00187 Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows Hiroataka Suetake; Tsukasa Fukusato; Takeo Igarashi; Tazro Ohta	

GigaScience

We would like to thank reviewers for their thoughtful comments and constructive feedbacks. We have added a summary figure and some additional paragraphs to make things clear, and made changes in the figures from the previous version of the manuscript.

Our responses and explanations to the issues pointed by each reviewer are follows.

Reviewer #1

> 1. Based on your survey of existing systems, could you possibly make a figure or table that showcases the features supported/not supported by these different systems, including yours?

Response: Thank you for the suggestion that can highlight the contribution of our work. We added the Table 2 in the Result section to compare the characteristics of WorkflowHub, Dockstore, nf-core, and a yevis-based registry. We compared them with three aspects; diversity, reliability, and usability of registered workflow.

> 2. Thoughts on security/cost safeguards? Perhaps beyond the scope, but it does seem like a governing group needs to define some limits to the testing resources and be able to enforce them. If I am a bad actor and programmatically open up 1000 PRs of expensive jobs, I'm not sure what would happen. Actions and artifact storage aren't necessarily free after some limit.

Response: We first had security concerns with the functionality in GitHub. Then, to solve problems of offensive actions by using malicious users via Pull Requests, we use GitHub's first-time contributor restriction. In this system, users (who first contribute to the repository) cannot perform a GitHub action without permission from the administrator. That is, the owners can easily reject "Pull Requests" from suspicious users, i.e., avoiding possible security issues.

As for the resource limitation of GitHub Actions, we solve the problem by using an external WES. In addition to a lack of resources, GitHub Actions may not be sufficient for testing workflows that only work on specific system configurations, such as GPUs or job schedulers. In such cases, the registry owner can check the testing portability by running the test on a remote WES instance. We added sentences to the Discussion section about the limitations of GitHub resources, the incident possibilities, and solutions to prevent them.

We added the following sentences to the Discussion section:

Automatic testing with GitHub actions may also cause the issue of computational resource shortage. To extend the capability of testing, Yevis has the option to specify the location of an external WES endpoint to run the test, which also enables the testing with a specific computational request such as GPUs or job schedulers.

> 3. What is the flow for simply updating to a new version of an existing workflow? (perhaps this could be in your docs, not necessarily this manuscript).

Response: We appreciate the comment about the version updates, which is one of the critical features of our implementation. The system identifies a workflow by its remote URL of the primary workflow in the metadata. Therefore, if the user modifies the metadata without changing the URL and bumps up the version information, the system recognizes that they are different versions of the same workflow. Users can use different versions of the workflow by specifying the TRS endpoint with a path including version information, such as "tools/0d2ae4c2-fe4c-48f7-811a-ac277776533e/versions/1.0.0".

Users can upgrade the registered workflow by performing the following steps.

1. Edit the existing yevis-metadata.yml file.
2. Bump the workflow version in the metadata.

3. Run tests using yevis-cli and send a PR to the repository.
4. Follow the same steps as adding a new workflow, and the new version will be published.

We added the above instruction to the online documentation.

> 4. CWL is an example of a workflow language that developers can extend to create custom "hints" or "requirements". For example, seven bridges does this in cavatica where a user can define aws spot instance configs etc. WDL has properties to config GCP images. It seems like in these cases, tests should only be defined to work when running "locally" (not with some scheduler/specific cloud env). But the author's do mention that tests will first run locally on the user's environment, so that does kind of get around this.

****Response:**** As we see the importance of the testing portability, we implemented the system with a strict specification. Developers can push the workflows that passed the local test to the registry, and the owners can merge those that passed the remote test. However, as the reviewer points out, specific compute-dependent tests exist in actual use cases and cannot be ignored. Allowing tests to be run outside the developer's own environment can solve the testing portability problem, i.e. the testing on a computing environment that anyone can access, even if not on GitHub. Therefore, the ability of our GitHub action to request tests to external WES instances can solve the problem. However, another problem is preparing a WES instance that can correctly run the tests for that workflow. We believe it is the responsibility of developers who want to share their workflows to prepare an environment where anyone can execute the tests correctly. Perhaps the preparation of a testing environment is something that the community should discuss.

> 5. For the "findable" part of FAIR, how possible is it to have "tags" of sort associated with a wf record so things can be more findable? I imagine when there is a large repository of many workflows, being able to easily narrow down to the specific domain interest you have could be helpful.

****Response:**** It is crucial to curate the metadata in the registry with tags or ontology terms that increase findability. The findability issue is a downside of the de-centralized registry model. Although it is possible to add tags as part of the TRS response specification, tagging with uncontrolled vocabulary by each workflow developer will not improve the findability. We believe that a centralized search index is in need to distinguish workflows from other similar or similar but different workflows available online. Therefore, a possible strategy is to register the metadata in a centralized registry such as WorkflowHub, while the workflow body is hosted and test runs on GitHub/Zenodo, as we proposed.

We have added this point as a weakness of the de-centralized model in the Discussion as following:

Another challenge for the proposed distributed registry model is the findability of workflows. In the model where each developer is responsible for their content, the use of appropriate terms for describing workflow metadata can be an issue. A possible solution to improve the findability of workflows in distributed registries is to collect metadata in a centralized registry to curate them and create the search index. However, this will require a further challenge to distinguish the collected workflows using only metadata.

Reviewer #2

Main concern

> I have one major gripe though, blocking acceptance: The choice to only support GitHub for hosting. There is a growing problem in the research world that more and more research is being dependent on the single commercial actor GitHub, for seemingly no other reason than convenience. Although GitHub to date can be said to have been a somewhat trustworthy player, there is no guarantee for the future, and ultimately this leaves a lot of research in an unhealthy dependenc on this single

platform. As a small note of a recent change, is the proposed removal of the promise to not track its users (see <https://github.com/github/site-policy/pull/582>).

> A such a central infrastructure component for research as a workflow registry has an enormous responsibility here, as it may greatly influence the choices of researchers in the future to come, because of encouragement of what is "easier" or more convenient to do with the tools and infrastructure available.

> With this in mind, I find it unacceptable for a workflow registry supporting open science and open source work to only support one commercial provider.

> The authors mention that technically they are able to support any vendor, and also on-premise setups, which sounds excellent. I ask the authors to kindly implement this functionality. Especially the ability to run on-premises registries is key to encourage research to stay free and independent from commercial concerns.

****Response:**** We understand the reviewer's concern about the lock-in risk to a commercial service well. It is not our desire to be locked into a specific service, and we are aware of the risk that these services (and every other service run by a vendor) may become unavailable to everyone. However, at the same time, it is impractical for us to implement and provide the "perfect lock-free" version of the system with production-level quality as well. We would like to explain the reasons below.

In this study, we aimed to achieve two major objectives: the independency from computational resources maintained by individual developers, and the automation to reduce the developers' tasks that make workflows reusable.

For the first objective, the academic communities know from their experience how difficult it is to maintain one's server for decades, at least for most of the developers in research. We think the main reason why many researchers rely on GitHub is simple, they do not need to maintain their git servers. This is also why we chose to use services run by vendors while realizing the risk of lock-in.

For the second objective, we had to retrieve the metadata of workflows automatically. Therefore, we used the GitHub API as a core component, which is not straightforward to replace with a plain on-premise git server.

Therefore, we have to give up these two strong benefits if we wanted to provide the same functionalities with a possible on-premise Yevis implementation. We still can think of the lock-free version of the registry, however, it will look like a completely different one as follows:

1. Workflow developers manually create metadata.yml and run the tests by hand using WES.
2. Developers submit the metadata to the registry administrator using a communication channel such as a mailing list.
3. The administrator runs tests using WES based on the received metadata.yml and reviews them.
4. Upload the reviewed workflow to any file server and publish it.

With these steps, partially supported by the current yevis-cli, one can have a workflow registry completely vendor lock-free. However, it is not as far efficient as the system using GitHub and Zenodo.

We believe our contribution is not the system itself but rather the distributed registry model in which communities can have their platforms by automating the quality assurance tasks. We hope that GitHub will not become evil, but if it does, we will re-implement our system with other services.

To clarify the benefit and the risk of vendor lock-in, and to explain the form of the system without those services, we added the sentences to the discussion section as follows:

*Yet we currently provide only the implementation that depends on those two services while it is technically possible to build a Yevis registry in an on-premise server. It is because the vendor-free system can only achieve part of our objectives to provide the effortless management of a registry. Here, we recognize a trade-off of building an

academic tool using services run by commercial vendors, which requires further discussion in the communities.*

Minor concerns

> 1. I think the manuscript is a missing citation to this key workflow review, as a recent overview of the bioinformatics workflows field, for example together with the current citation [6] in the manuscript:

> Wratten, L., Wilm, A., & Göke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature methods*, 18(10), 1161-1168.

> <https://www.nature.com/articles/s41592-021-01254-9>

****Response:**** Thank you for pointing out the important citation. We added the citation as suggested.

> 2. Although it might not have been the intention of the authors, the following sentence sounds unnecessarily subjective and appraising, without data to back this up (rather this would be something for the users to evaluate):

> The Yevis system is a great solution for research communities that aim to share their workflows and wish to establish their own registry as described.

> I would rather expect wording similar to:

> "The Yevis system provides a [well-needed] solution for ..."

> ... which I think might have been closer to what the authors intended as well.

****Response:**** We thank the precise suggestion to the statement. We edited the sentence as suggested.

Reviewer #3

> One of my main concerns is that this work presents a technical implementation but lacks an architecture diagram or a big picture that would clarify the contributions of this work with respect to the features provided by the external platforms (GitHub, Zenodo). This would also help the reader understand how the solution proposed can be reused with possibly other services.

****Response:**** We agree with the reviewer's concern that the system's overall architecture is not clear to the readers. Although we expected that the Figure 1 in the original submission shows the overall procedure with the components we used, it focuses on the steps of the submission, not the environment where the system is running. We added a new diagram as Figure 1 to indicate the components where each process is running and how they are connected each other.

> It is also hard to understand the technical solutions when target users and their typical needs have not been clearly stated beforehand. It seems to be very complicated for non-developers to use or operate Yevis, especially when testing workflows through the "GitHub actions" infrastructure. Is the Yevis platform only targeting workflow developers? It was not easy to understand the benefits offered to research communities aimed at sharing/reusing workflows.

****Response:**** Firstly, we would like to thank the reviewer for pointing out that it is not clear about the system's target user is. We agree that it was implicit in the manuscript that the technical level of the users expected to use the system. We expect workflow developers as the target users of the Yevis system, i.e., developers and researchers with the skills to design and implement arbitrary workflows using the workflow language. However, the procedure for publishing workflows using the Yevis system does not require any of the difficulties noted by the reviewer. The repository template used in the Yevis setup procedure already has the GitHub action for workflow testing that the system automatically executes when a pull request is received. Thus, the users do not need to write any GitHub actions but prepare the test data for their workflows and write their metadata. As indicated in the new system overview figure, many procedures exist to publish. However, the system automates them except for the writing of metadata and the administrator's review. We believe that this labor-saving automation is the contribution of this research.

To clarify the target users, we added a sentence to the Background section as follows:

The system expects developers and researchers who design workflows using workflow languages as users, although it does not require advanced computer skills to operate the system.

> Regarding the background section, GA4GH-TRS is not introduced while mentioned as part of the results. It's hard for the reader to understand how the use of this standard contributes to better workflow sharing (metadata ?) or better reuse (tests?).

****Response:**** We added the sentence to the second paragraph of the System design section as follows:

The TRS defines the specification of computational tool/workflow metadata representation, including workflow's URI, used language, version, etc. It ensures the interoperabilities among different tool/workflow registries and enables workflow engines to retrieve the information to execute a workflow maintained at a remote server, which improves reusability of published workflows.

> Other workflow registries such as WorkflowHub or NF-Core have been described in the background section but a dedicated state-of-the-art section would have allowed the authors to provide more details on the positioning of Yevis. Some related works should also be part of the analysis such as BIAFLOWS also providing a benchmarking environment, or OpenEBench.

****Response:**** We added the Table 2 to compare the existing registries, including the WorkflowHub, nf-core, and Dockstore, to compare their characteristics with a Yevis-based registry. We reviewed the BIAFLOWS and OpenEBench and we recognized their features of tool/workflow collections and evaluations. However, their objectives are to provide a summary to compare different computational tools for better tool selection, which is a slightly different scope from the aim of this study to publish workflow with testing results for better trust. Therefore, we exclude them from the table, but we thank the reviewer for pointing out the relevance of our system with those excellent community resources.

> The live deployment URL of the Yevis system should be provided in the paper so that readers can browse/reuse already registered workflow. The link provided in the source code repository only shows 4 workflows and not the DAT2 workflow used in the "proof of concept" section. Is the system limited to some workflow engines ? Would it be possible to register and test Galaxy workflows for instance ?

****Response:**** In the original manuscript, we mentioned the registry URL of the DAT2-cwl registry in the reference section (37), but we recognized it was not clear to the readers. We added the URL to the DAT2-cwl registry (<https://github.com/pitagora-network/yevis-DAT2-cwl>) to the "sharing workflows using Yevis" section and the repository of yevis-cli. The registry with four workflows is the DDBJ workflow registry, which is different from the one we have shown as a demo in the manuscript. We also have corrected the resources mentioned in the manuscript.

The yevis system itself does not have a limitation on workflow engines but requires a WES endpoint to test the given workflow. Thus, if we had a WES instance that consumes a Galaxy workflow the system can accept a Galaxy workflow. Currently, we have a WES instance available for Common Workflow Language, Nextflow, Workflow Description Language, and Snakemake.

> Regarding tests of workflows, the files associated to HiSAT2 on the Pitagora workflow were not accessible (404 not found). I also had some difficulties when trying to inspect the results of the automated tests in the GitHub actions. For this workflow, [Add workflow: Pitagora CWL - Download SRA · ddbj/workflow-registry@89961a4 · GitHub](<https://github.com/ddbj/workflow-registry/actions/runs/2256980244>), the logs were expired and thus no more accessible. This highlights the challenge of relying on external computing services to run possibly long and costly executions, even with test data.

****Response:**** We thank the reviewer for pointing out a critical issue of the current implementation. We missed this end-of-life of the GitHub actions artifact during our development phase. This GitHub action artifact is for two objectives, and we expect this limitation does not affect the current Yevis procedure. First, the artifact is supposed for the review by the registry administrator. If 90 days have passed since the automatic execution and the test log is missing, the admin can re-run the action and review the artifact again. Second, the artifact is also valuable as a workflow run provenance. For this purpose, we have added a function in the WES instance to generate ResearchObject-crate (RO-crate), a standard specification to record the research artifact, to let the Yevis system store the test log in the Zenodo repository.

For long and costly workflow execution problem, the system has an option to run tests with the external WES endpoint shown in the new summary figure. It brings the availability issue of the testing instance but can solve the problem of the resource limitation of the GitHub actions.

> Regarding the validation of metadata, very few informations are provided. Are all metadata fields mandatory ? are some fields recommended ? Which kind of validation is performed ? How the result of validation is returned to users ? Regarding the metadata themselves, how do they comply with community emerging standards such as Bioschemas or RO-crate ? At the time of the review, it was not possible to find any semantic annotations in the Yevis web page, thus limiting the discoverability and the interoperability of workflows descriptions.

****Response:**** For each field in the metadata, the system performs a type check, license check, URL validation, remote file existence, etc. We added the information on these validation items to the CLI's GitHub repository. (<https://github.com/sapporowes/yevis-cli#validate>) The CLI checks these metadata items written by the user, and if it finds an invalid field, the CLI outputs an error message and prompts the user to correct it. As the reviewer points out, our specification for workflow metadata overlaps with some of the community-developed standards. Currently, for workflow test run logs, we are working with the community to apply the workflow run crate under development by the RO-crate community. However, as we prioritize the readability and editability of Yevis metadata, the approach could be to adopt a proprietary format and then convert it to a standard by the community.

Because of technical limitations in the current implementation, embedding tags in the pages is not practical to improve findability. As we implemented the Yevis-web in the JavaScript Single Page Application framework, it is not possible to embed different annotation tags in the HTML source of each workflow page.

However, we believe workflow findability should not be left to individual workflow developers but should be a centralized effort by a central registry. Therefore, we would like to enhance the findability of workflows by having individual communities manage their workflows with Yevis while a centralized registry such as WorkflowHub collects these metadata.

> Finally, the discussion and future works sections could be enriched to address for instance.

> - the scalability of the approach with possibly long or costly tasks when testing workflows

> - the interoperability of this platform with other registries

> - the genericity of the approach (is it applicable in the context of other scientific disciplines)

> - the use of this platform to compare or benchmark workflow executions based on predefined test datasets

****Response:**** We would like to thank the reviewer's suggestion to make the Discussion section more constructive.

We added the following topics in the Discussion section to reflect the reviewers' feedback.

- Using external WES endpoint to overcome the resource limitation of GitHub actions:

also brings the concern of availability (3rd paragraph)
- The automation of quality control procedure using CI/CD of the public services: its limitation and possibility (3rd paragraph)
- The interoperability of the Yevis registry and possibilities of coexistence with the other centralized registries for findability (6th paragraph)

We excluded the topic of benchmarking because we think it can confuse the readers as if the Yevis system can use for that purpose. We expect benchmarking feature should be provided on a different layer than that of the registry.

Minor comments

> Figures 2 and 7 seems to be very similar. Only one should be kept.

****Response:**** We agree with the reviewer's assessment. To avoid the confusion, we removed the Figure 7 which showed the metadata file modified by the yevis system. We remained Figure 2 and added the explanation that the metadata will be modified.

> How are test specified, is the specification generic enough? How does it support multiple workflow engines?

****Response:**** Yevis's specification of the test run conforms to the GA4GH WES specification. In yevis-cli, the option --wes-location specifies the WES instance on which the system runs the test. As WES implementations usually have a single workflow engine, users need to use a WES capable of executing the workflow's language. As an exception, Sapporo, one of the WES implementations we have developed, provides users the option to specify a workflow engine from several engines covering CWL, WDL, Nextflow, and Snakemake. The yevis-cli command uses Sapporo via Docker if users have no WES preference.

We added sentences to emphasize the option to specify an external WES for testing instance as follows:

Automatic testing with GitHub Actions may also cause the issue of computational resource shortage. To extend the capability of testing, Yevis has the option to specify the location of an external WES endpoint to run the test, which also enables the testing with a specific computational request such as GPUs or job schedulers.

> The paragraph on decentralization in the discussion is confusing. All workflow executions seem to be centralized on the GitHub infrastructure with no control on data or computation placement. The same happens for data on the Zenodo infrastructure.

****Response:**** Thank you for your comments. We think there is a misunderstanding about the term decentralization. In the submission stage, we described "Yevis can promote the concept of a distributed workflow registry model that underlies the specifications of the GA4GH Cloud Work Stream," so this paragraph claim is the idea of decentralization of workflow registries, not computation placement. Since we emphasize that the communities have an option to build their own registry, not relying on a centralized registry such as WorkflowHub or nf-core, we added a sentence to clarify the idea as follows:

In the distributed workflow registry model, researchers have the option to build their own workflow registry, rather than submitting to a centralized registry.

As mentioned above, the computation also can be decentralized using the external WES instances.

> DAT2-cwl is not registered in the live deployment of Yevis

****Response:**** The original manuscript has referred external GitHub repositories and the resources, which may confuse readers. We added the URL in the sentence along with the citation to lead the readers to the appropriate resource.

Additional Information:

Question	Response
<p>Are you submitting this manuscript to a special series or article collection?</p>	<p>No</p>
<p>Experimental design and statistics</p> <p>Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available in the figure legends.</p> <p>Have you included all the information requested in your manuscript?</p>	<p>Yes</p>
<p>Resources</p> <p>A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible.</p> <p>Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?</p>	<p>Yes</p>
<p>Availability of data and materials</p> <p>All datasets and code on which the conclusions of the paper rely must be either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript.</p>	<p>Yes</p>

Have you have met the above requirement as detailed in our [Minimum Standards Reporting Checklist?](#)

Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows

Hirota Suetake¹, Tsukasa Fukusato², Takeo Igarashi¹, and Tazro Ohta³✉

¹Department of Creative Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

²Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

³Database Center for Life Science, Joint Support-Center for Data Science Research, Research Organization of Information and Systems, Shizuoka, Japan

Background: Many open-source workflow systems have made bioinformatics data analysis procedures portable. Sharing these workflows provides researchers easy access to high-quality analysis methods without the requirement of computational expertise. However, published workflows are not always guaranteed to be reliably reusable. Therefore, a system is needed to lower the cost of sharing workflows in a reusable form.

Results: We introduce Yevis, a system to build a workflow registry that automatically validates and tests workflows to be published. The validation and test are based on the requirements we defined for a workflow being reusable with confidence. Yevis runs on GitHub and Zenodo and allows workflow hosting without the need of dedicated computing resources. A Yevis registry accepts workflow registration via a GitHub pull request, followed by an automatic validation and test process for the submitted workflow. As a proof of concept, we built a registry using Yevis to host workflows from a community to demonstrate how a workflow can be shared while fulfilling the defined requirements.

Conclusions: Yevis helps in the building of a workflow registry to share reusable workflows without requiring extensive human resources. By following Yevis's workflow-sharing procedure, one can operate a registry while satisfying the reusable workflow criteria. This system is particularly useful to individuals or communities that want to share workflows but lacks the specific technical expertise to build and maintain a workflow registry from scratch.

Workflow | Workflow language | Continuous integration | Open science | Reproducibility | Reusability

Correspondence: t.ohta@dbcls.rois.ac.jp

Background

Due to the low cost and high throughput of measurement instruments that acquire digital data from biological samples, the volume of readily available data has become enormous (1). To obtain scientific knowledge from large datasets, a number of computational data analysis processes are required, for example, in DNA sequencing, sequence read trimming, alignment with reference genomes, and annotation using public databases (2). Researchers have developed analysis tools for each process and often publish them as open-source software (3). To avoid the need to execute these tools manually, researchers usually write a script to combine them into what is called a workflow (4).

To build and maintain a complex workflow that combines many tools efficiently (5), many workflow systems have been developed (6, 7). Some of these systems have large user communities, such as Galaxy (8), the Common Workflow Language (CWL) (9), the Workflow Description Language (WDL) (10), Nextflow (11), and Snakemake (12). Although each system has its unique characteristics, they have a common aim: to make computational methods portable, maintainable, reproducible, and shareable (4). Most systems have a syntax for describing a workflow that is part of what is called a workflow language. They also have an execution system that works with computational frameworks, such as a job scheduler and container virtualization (13).

With the popularization of workflow systems, many research communities have worked on workflow sharing in the form of a workflow language. Workflow registries, such as WorkflowHub (14), Dockstore (15), and nf-core (16), have been developed as public repositories for the sharing of workflows. Workflow execution systems also utilize these registries as their tool libraries. To improve the interoperability of workflow registries, the Global Alliance for Genomics Health (GA4GH) proposed the Tool Registry Service (TRS) specification that provides a standard protocol for sharing workflows (17, 18).

Sharing workflows not only increases the transparency of research but also helps researchers by facilitating the reuse of programs, thereby making data analysis procedures more efficient. However, workflows that are accessible on the internet are not always straightforward for others to use. If a published workflow is not appropriately licensed, researchers cannot use it because the permission for secondary use is unclear. A workflow may also not be executable because its format is incorrect, or dependent files cannot be found. Even if a workflow can be executed, the correctness of its operation often cannot be verified because no tests have been provided. Furthermore, the contact details of the person responsible for the published workflow are not always attached to it.

It is noteworthy that these issues in reusing public workflows are not often obvious to workflow developers. To clarify the requirements for workflow sharing, Goble et al. have proposed the concept of a FAIR (findable, accessible, interoperable, and reusable) workflow (19). This inheritance of the FAIR principles (20) focuses on the structure, forms, ver-

sioning, executability, and reuse of workflows. However, the question remains as to who should guarantee to users that published workflows can be reused following the FAIR workflow guidelines.

WorkflowHub asks submitters to take responsibility for workflows: when a workflow is registered on WorkflowHub, the license and author identity should be clearly stated, encouraging them to publish FAIR workflows. However, there is no obligation as to the correctness of the workflow syntax, its executability, or testing. Not placing too many responsibilities on workflow submitters keeps obstacles to submission low, which will likely increase the diversity of public workflows on WorkflowHub; however, it will also likely increase the number of one-off submissions, which one can assume are at higher risk for the workflow problems previously described.

Unlike WorkflowHub, in *nf-core*, the community that operates the registry holds more accountability for published workflows. Workflow submitters are required to join the *nf-core* community, develop workflows according to their guidelines, and prepare them for testing. These requirements enable *nf-core* to collect workflows with remarkable reliability. However, the community's effort tends to focus on maintaining more generic workflows that have a large number of potential users. Consequently, *nf-core* states that it does not accept submissions of bespoke workflows. This is an understandable policy, as maintaining a workflow requires domain knowledge of its content, and this is difficult to maintain in the absence of the person who developed the workflow.

In order to improve research efficiency through workflow sharing, research communities need the publication of diverse workflows in a reusable form. However, as shown by existing workflow registries, there is a trade-off between publishing a wide variety of workflows and maintaining the reusability of the workflows that are published. Solving this issue requires reducing the cost to developers in making and keeping their workflows reusable, which currently relies on manual effort. This is achievable by redefining the FAIR workflow concept as a set of technical requirements and providing a system that automates their validation and testing.

We introduce *Yevis*, a system to share workflows with automated metadata validation and test execution. The system expects developers and researchers who design workflows using workflow languages as users, although it does not require advanced computer skills to operate the system. Through the development of *Yevis*, we specified a set of technical requirements that define a reusable workflow, according to the FAIR workflow concept. *Yevis* helps researchers and communities share workflows that satisfy the requirements by supporting a build of an independent workflow registry. To allow workflow hosting without the need of additional dedicated computing resources, *Yevis* works on two public web services: GitHub, a source code sharing and software development service, and Zenodo, a public research data repository. In addition, a *Yevis* registry provides a web-based workflow browser and the GA4GH TRS-compatible API ensures interoperability with other existing workflow registries. *Yevis* is

particularly powerful when individuals or communities want to share workflows but are without the technical expertise to build and maintain a web application. To demonstrate that workflows can be shared that fulfill the defined requirements using *Yevis*, we built a registry for workflows that an existing community has managed.

Implementations

System design. Figure 1 shows the overall architecture of the workflow registry built by *Yevis*. The repository administrator uses our GitHub repository template and follows the guide to set up a *yevis*-based registry creating new repositories on GitHub and Zenodo. After creating the metadata and passing the workflow test on a local computer, workflow developers submit the metadata as a pull request to the GitHub repository. Once the repository receives the pull request, it automatically tests the workflow again on GitHub Actions, GitHub's continuous integration/continuous delivery (CI/CD) environment. The system has the option to use an external WES instance for testing before accepting the submission. The registry administrator will check the test result and approve, that is, merge the pull request. Once the submission is approved, the repository runs another GitHub Actions automatically to upload the content to the Zenodo repository and the GitHub pages.

To implement the system, we first defined a set of requirements that the *Yevis* system can automatically verify or test (Table 1). By satisfying these requirements inspired by FAIR workflow, we consider a workflow is "reusable with confidence." These criteria have three aspects: workflow availability, accessibility, and traceability. The TRS defines the specification of computational tool/workflow metadata representation, including workflow's URI, used language, version, etc. It ensures the interoperabilities among different tool/workflow registries and enables workflow engines to retrieve the information to execute a workflow maintained at a remote server, which improves the reusability of published workflows. To help researchers share reusable workflows, we took an approach to aid them in building their own workflow registry that automatically ensures its reusability. We define a workflow registry as a service that serves workflow information via the GA4GH TRS protocol.

The information provided by the TRS API is various workflow metadata, such as author information, documentation, language type and version, dependent materials, testing materials, etc. Large files, such as dependent materials and testing materials, are not directly included in the TRS API response but are described as remote locations, such as HTTP protocol URLs. Therefore, the entities that a workflow registry collects are a set of workflow metadata described in the form of the TRS API response. In this study, therefore, we designed the system as an API server that delivers the TRS API response.

In the *Yevis* registry, a workflow-sharing procedure is divided into three processes: submission, review, and publication (Figure 2). To address the requirements listed in Table 1, the *Yevis* system automatically performs processes, such as

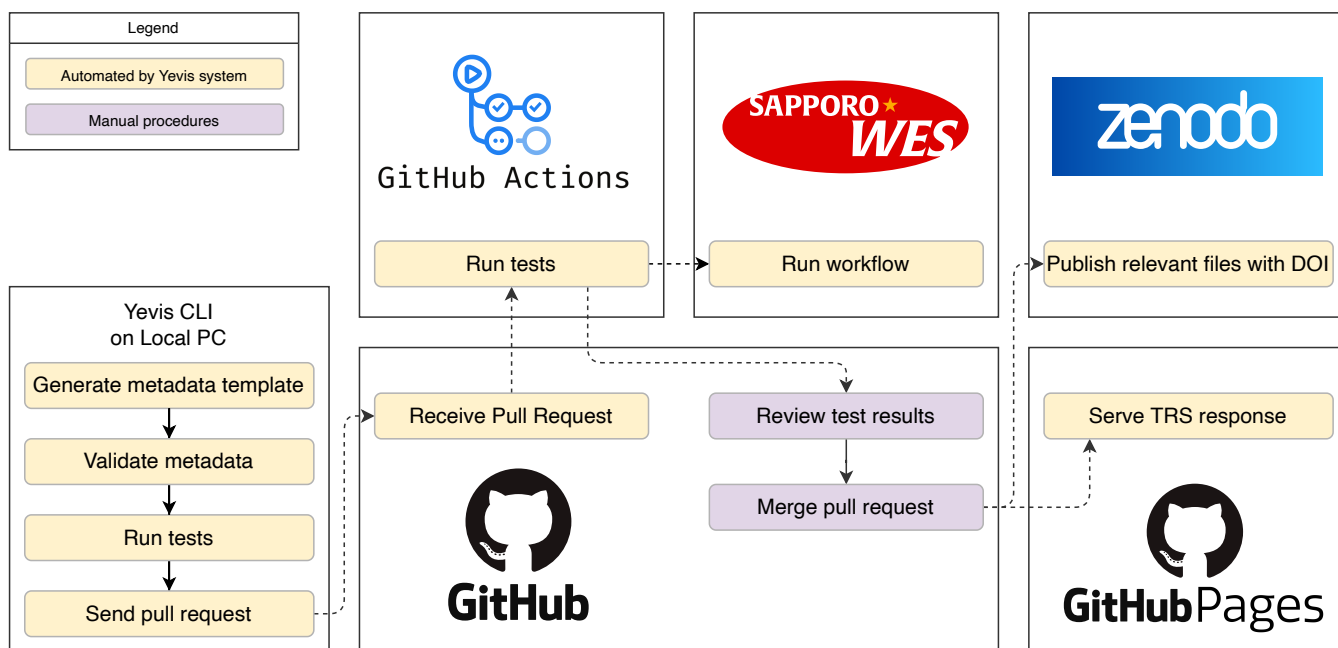


Fig. 1. The overall architecture of the yevis system. The registry administrator needs to set up a GitHub repository from our repository template and a Zenodo repository for file persistence. Workflow developers test their workflows on a local computer using our Yevis-cli, then submit a pull request to the GitHub repository. The GitHub repository has two GitHub Actions, testing on GitHub Actions or an external WES instance, and publishing workflow contents and metadata to the Zenodo repository and GitHub pages.

Table 1. The requirements for a workflow to be considered reusable with confidence. We classify these requirements from the perspectives of the availability, validity, and traceability of the workflows. We propose that these requirements should be assured and provided to users by the workflow registries.

Perspective	Requirement	Description
Availability	Main workflow description	The main workflow description file is available and accessible without restriction.
	Dependent materials	The dependencies of the main workflow are available, e.g., definitions of dependent workflows and tools.
	Testing materials	The job configuration files for testing are available, e.g., parameter and input files.
	Open-source license	The workflow and the related materials are published under an appropriate open-source license.
Validity	Language type	The language used to describe the workflow is specified, e.g., CWL, WDL, or Nextflow.
	Language version	The version of the workflow language used is specified.
	Language syntax	The language syntax of the workflow is valid.
Traceability	Authors and maintainers	The contact information of the authors and the maintainers is identified.
	Documentation	The documentation of the workflow is available.
	Workflow ID	The unique identifier to specify the workflow is assigned, ideally by a URI.
	Workflow metadata version	The version number of the workflow metadata is specified.

metadata validation, workflow testing, test provenance generation, persisting associated files, DOI assignment, and TRS response deployment. To generate the TRS API response and publish it while addressing the requirements listed in Table 1, we implemented a command-line application called Yevis-cli. This application contains various utilities to support the workflow registration procedure including validation and testing. As a service and infrastructure to perform these steps, we designed Yevis to use the services of GitHub and Zenodo. Using these web services makes it possible for communities to build a workflow registry without the need of

maintaining their own computer servers.

Workflow registration with automated validation and testing. To set up a Yevis registry, registry maintainers need to do an initial configuration of GitHub and Zenodo; this involves, for example, creating a GitHub repository, changing repository settings, and setting up security credentials. The online documentation “Yevis: Getting Started” shows the step-by-step procedures to deploy a workflow registry and test it (21).

We defined the Yevis metadata file, a JSON or YAML format

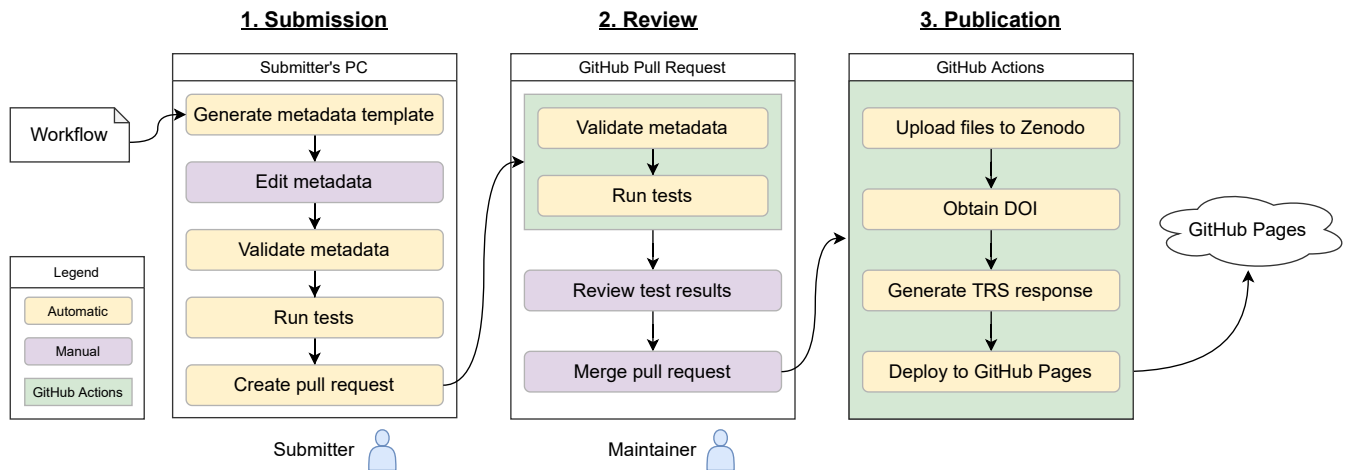


Fig. 2. The flowchart of the workflow registration to a Yevis repository. The workflow registration procedure is divided into three processes: the submission, review, and publication process. Each process is performed in different locations: in the submitter's local environment, as part of the GitHub pull request, or as the GitHub Actions. The generated TRS API response is deployed to GitHub Pages. The steps indicated by yellow boxes, such as validating metadata, are performed automatically using Yevis-cli.

file that contains structured workflow metadata (Figure 3). Yevis-cli uses this file as its input and output in the submission process. The Yevis metadata file will be published on the registry along with the TRS response to provide metadata that is not included in the TRS protocol, such as an open-source license.

Submission process. Figure 4 shows the submission process using Yevis-cli. During this submission process, the workflow submitter describes the workflow metadata in their local environment and submits it through a GitHub pull request (i.e., a review request to the registry maintainer). First, Yevis-cli generates a template for the Yevis metadata file, which requires the URL of the main workflow description file as an argument. In many workflow systems, the main workflow description file is the entry point for workflow execution. Yevis-cli generates a template supplemented with workflow metadata automatically collected by using the GitHub REST API and inspecting the workflow's contents. Next, the submitter needs to edit the Yevis metadata file template and add workflow tests. Yevis-cli executes a test using a GA4GH Workflow Execution Service (WES) instance, a type of web service also described as workflow as a service (18, 22); therefore, the testing materials must be written along with the specification of the WES run request. Yevis-cli performs these tests to check if the workflow execution completes successfully. After preparing the Yevis metadata file, Yevis-cli validates the workflow metadata syntax and runs tests using WES in the submitter's local environment. If no WES endpoint is specified, the tests are run using Sapporo (23), a production-ready implementation of WES, and Docker (24), a container virtualization environment. Using these portable WES environments also ensures the portability of testing in Yevis. Finally, Yevis-cli submits the workflow as a GitHub pull request, once it confirms the required actions: the metadata validation and the test passing. This restriction reduces the burden on the registry maintainer because many of the requirements listed in Table 1 can be ensured during the submission process rather than the review process.

Review process. Figure 5 shows the workflow review process using Yevis-cli. During the review process, registry maintainers examine each workflow submitted as a Yevis metadata file on the GitHub pull request UI. Because the submission method is restricted to Yevis-cli, the submitted workflow is guaranteed to pass validation and testing. To ensure the reproducibility of test results on a local computer, Yevis automatically validates and tests it on GitHub Actions (25). After automated validation and testing, the maintainers review the test results and log files to consider whether to approve the pull request. Rather than using a chat tool or a mailing list, the review process through the GitHub pull request improves the transparency and traceability of workflow publication.

Publication process. Figure 6 shows the workflow submission process using Yevis-cli. During the publication process, the system automatically persists all files associated with the workflow. It generates the TRS response from the Yevis metadata file. The approval of the pull request automatically triggers the publication process on GitHub Actions. In the GitHub Actions script, Yevis-cli uses the Zenodo API to create a new Zenodo upload and persists all files related to the workflow (26). It obtains the DOI and persistent URLs of workflows from Zenodo, and appends them to the Yevis metadata file. Following the Zenodo upload, the Yevis-cli in the GitHub Actions generates a TRS response JSON file and is deployed to GitHub Pages, GitHub's static web page hosting service. Accordingly, the Yevis metadata file is merged to the default branch of the GitHub repository and deployed to GitHub Pages. With these two files, the TRS response JSON file and the Yevis metadata file, a Yevis registry covers the information that fulfills the requirements of a reusable workflow.

Workflow browsing interface. To make it easier for registry maintainers and users to browse workflows, we implemented Yevis-web, a workflow browsing interface (Figure 7). As the interface is a browser-based application implemented in JavaScript, registry maintainers can deploy the browser on

```

id: d03458d8-837c-4173-afa3-55ebe538b0b2
version: 1.0.0
license: Apache-2.0
authors:
  - github_account: suecharo
    name: "Suetake, Hirotake"
    affiliation: The University of Tokyo
    orcid: 0000-0003-2765-0049
workflow:
  name: DAT2-cwl - bacteria genome workflow
  readme: "https://raw.githubusercontent.com/pitagora-network/DAT2-cwl/main/workflow/bacteria-genome/README.md"
  language:
    type: CWL
    version: v1.0
  files:
    - url: "https://raw.githubusercontent.com/pitagora-network/DAT2-cwl/main/workflow/bacteria-genome/bacteria-genome.packed.cwl"
      target: bacteria-genome.packed.cwl
      type: primary
  testing:
    - id: test
      files:
        - url: "https://github.com/pitagora-network/DAT2-cwl/blob/main/workflow/bacteria-genome/test-job-yevis.yml"
          target: test-job-yevis.yml
          type: wf_params
        - url: "https://github.com/pitagora-network/DAT2-cwl/blob/main/test/data/DRR024501_1.fastq"
          target: DRR024501_1.fastq
          type: other
        - url: "https://github.com/pitagora-network/DAT2-cwl/blob/main/test/data/DRR024501_2.fastq"
          target: DRR024501_2.fastq
          type: other

```

Fig. 3. Example of the Yevis metadata file. The main workflow description, dependent materials, etc. are described as remote locations; the file contains all the information that the Yevis-cli requires to automate the whole process. This is the actual metadata file for the workflow described in the Section “Sharing workflows using Yevis.” This file is automatically updated through the processes within Yevis; for example, the file URL field is replaced by the Zenodo record URL that persists in the associated workflow files.

GitHub Pages. Yevis-web accesses the TRS API served via GitHub Pages and the GitHub REST API to retrieve workflow information. To help organize the submissions to the registry, the browser shows workflows of both statuses, those already published and those still under the review process.

Results

Feature comparison with existing registries. To clarify the advantages of a workflow registry built by Yevis, we compared the characteristics of a Yevis-based registry with WorkflowHub (14), Dockstore (15), and nf-core (16). As comparison views, we focused on three aspects; diversity, reliability, and usability of workflows available in a registry.

In the diversity of registered workflow, as “Acceptable workflows” in Table 2, WorkflowHub and Dockstore have an advantage because they have no restrictions on workflows in terms of their purposes or languages. As mentioned in the Introduction section, nf-core has the policy to collect only best-practice workflows written in Nextflow. In contrast, a Yevis-based registry can accept any workflows written in any language as long as the registry administrator approves the submission. The only limitation in a Yevis-based registry is the testing environment because the submission to the registry requires a suitable testing environment for the given workflow. By default, Yevis uses Sapporo WES for its test execution, a WES implementation with multi-engine support

which enables developers to extend its execution capability. With the reliability of available workflows, we prioritize the features such as general quality control of submissions and testing preparation. As shown in Table 2, in WorkflowHub and Dockstore, each developer is responsible for quality control and testing for the submission. As a result, they may have workflows that are not reusable, such as those lack dependencies, documentation, or the appropriate open-source license. The platforms do not have a strict testing policy, although it helps lower the barrier to submission. On the other hand, nf-core does quality control and testing of its workflows by its community to provide reliable workflows. In a Yevis-based registry, the registry itself provides automated functions to manage the quality of workflows based on the proposed requirements and test workflows in the submitter’s environment and the remote CI/CD environment.

For usability, we focused on two standardized forms to identify the workflow: DOI and TRS 2. A Yevis-based registry is only one of the four that provides DOI for each registered workflow. Assigning DOI for workflow files prevents the problem of altering resource URLs. For TRS compatibility, currently, nf-core is the only one not providing TRS responses. It may be because of the design of Nextflow language, which boosts developers’ productivity on a specific directory structure rather than using distributed relevant workflow files. However, three out of four has TRS compatibility, which helps data scientist write a tool to reuse the available

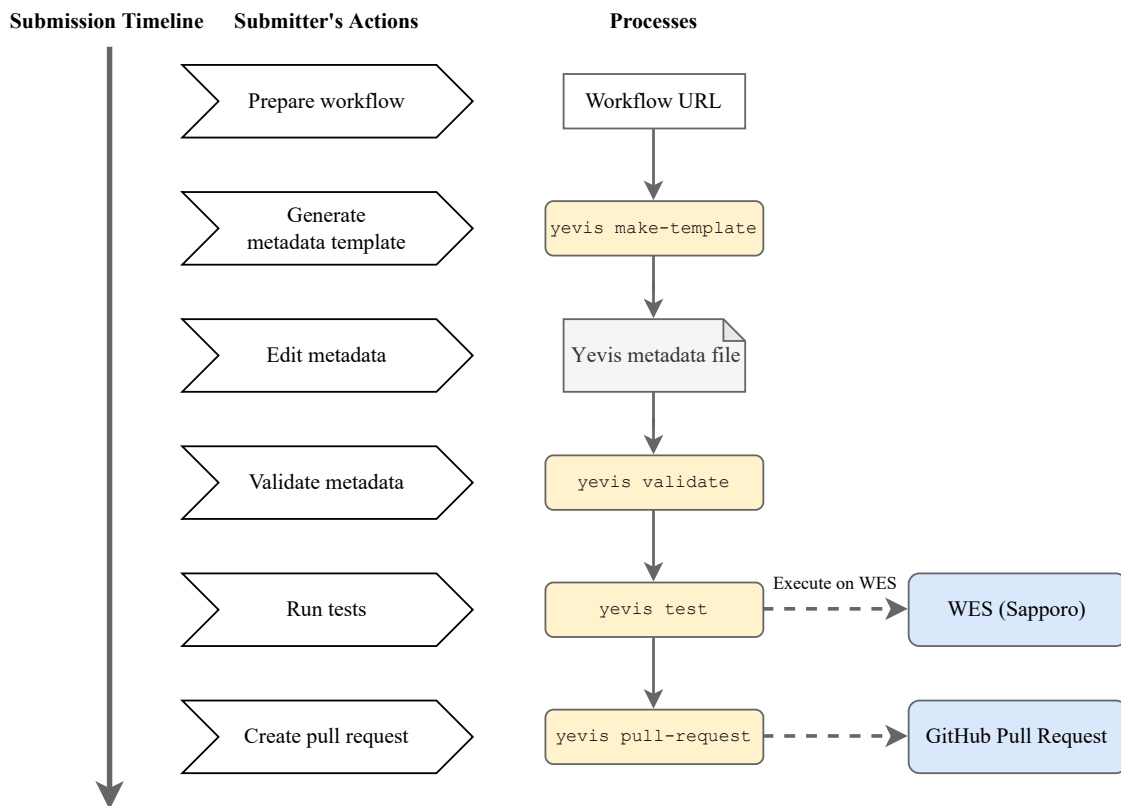


Fig. 4. The timeline of the workflow submission process using Yevis-cli. The submitter executes four subcommands of Yevis-cli: “make-template,” “validate,” “test,” and “pull-request” in its local environment. The submitter needs to edit a template of the Yevis metadata file using any text editor. The workflow and its metadata need to pass validation and testing before their submission, which helps to reduce the burden on the registry maintainer.

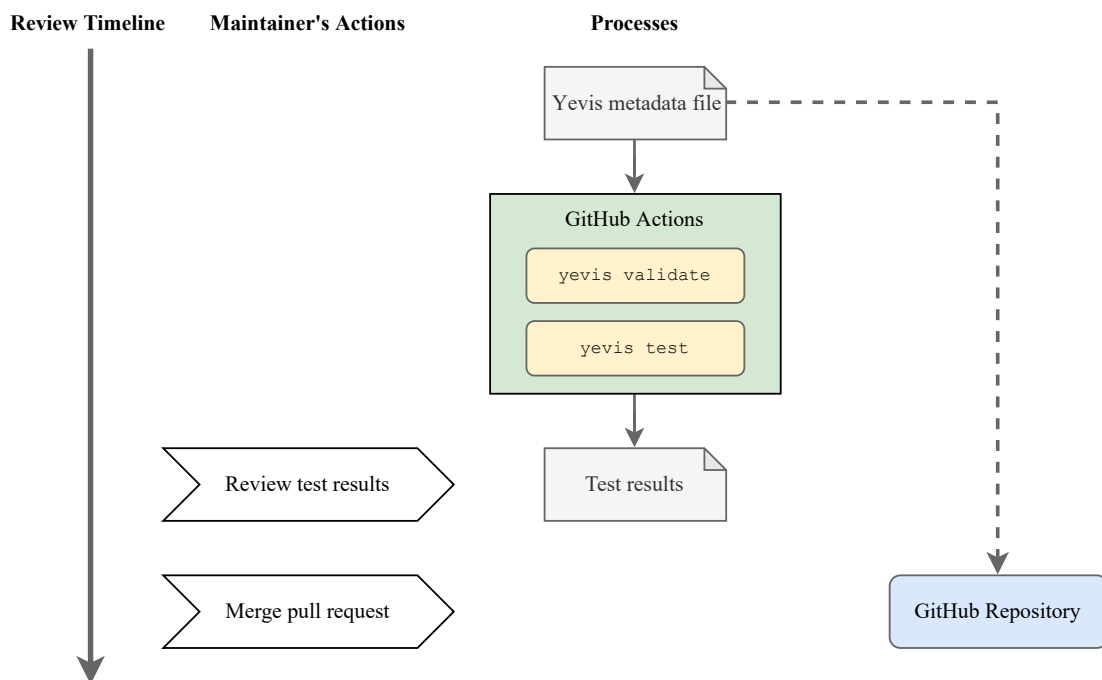


Fig. 5. The timeline of the workflow review process using Yevis-cli. The workflow and its metadata are again validated and tested automatically on GitHub Actions. The test results and logs can then be reviewed by the registry maintainers with the GitHub pull request UI.

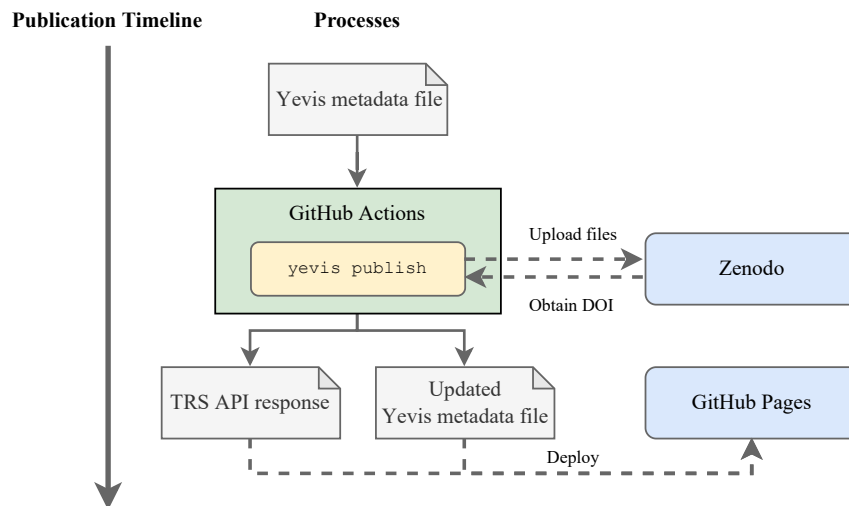


Fig. 6. The timeline of the workflow publication process using Yevis-cli. All steps are performed automatically on GitHub Actions. All files related to the workflow are persisted by uploading them to Zenodo. The DOI is generated by Zenodo, and the Yevis metadata file is updated to append the DOI information and the persisted file URL. The GitHub Actions generates a TRS response from the Yevis metadata; it then deploys both of them to GitHub Pages.

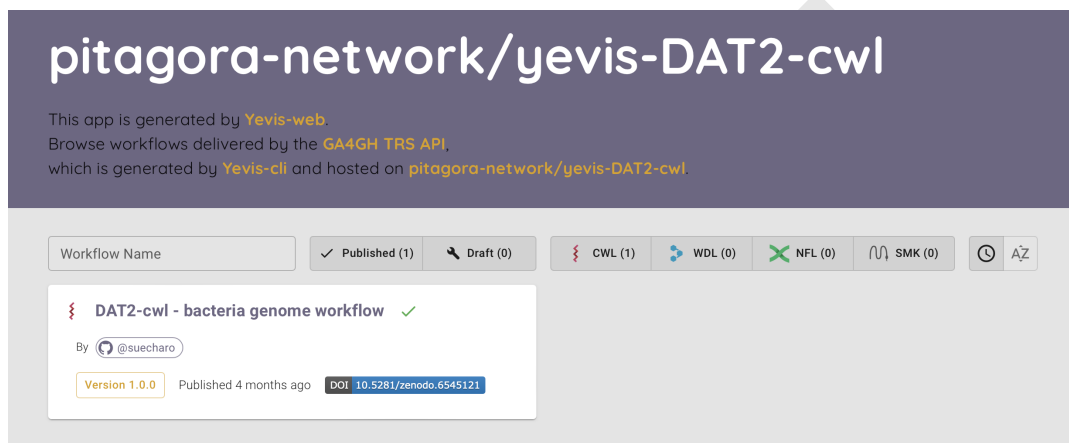


Fig. 7. Screenshot of Yevis-web. Yevis-web is a browser-based application used via a web browser, which is deployed by workflow registry maintainers and communicates with the TRS API and GitHub REST API to retrieve workflow information. The browser shows both published and under-review workflows to help maintainers in organizing the registry. Upon selecting a workflow of interest, Yevis-web displays more detailed information, such as test results and the contents of the files related to the workflow.

workflows with the unified API response.

Sharing workflows using Yevis. To demonstrate that a research community can publish the workflows using Yevis while addressing the requirements listed in Table 1, we built a workflow registry that publishes “DAT2-cwl” workflows with the Yevis system (27)¹. These workflows written in CWL are the appendix of the book *Next Generation Sequencer DRY Analysis Manual, 2nd Edition* (28) and are maintained by the book’s authors and communities. These workflows have been maintained by a community of bioinformatics experts; however, they fulfill only a part of the requirements that we defined. For example, the workflows have test data but would require continuous testing. They also lack workflow metadata in a standard format.

Among the DAT2-cwl workflows, we selected a bacterial

genome analysis workflow in building a new registry with Yevis (29). This workflow combines the following command line tools: SeqKit (30), FastQC (31), fastp (32), and Platanusb (33). Each tool used in the workflow is packaged in a Docker container. First, we described a Yevis metadata file (Figure 3) for this workflow using Yevis-cli and appended a test of the workflow in the form of a WES run request. We then performed the workflow registration procedure described in the Section “Workflow registration with automated validation and testing” using Yevis-cli that enable the automation of many of the steps in the validation, testing, reviewing, and publishing.

Through the publication procedure of the bacteria genome analysis workflow, we evaluated how the Yevis system addressed the requirements listed in Table 1. Requirements classified as “Availability” were addressed by being uploaded to Zenodo under an appropriate open source license (34). The Yevis metadata file (Figure 3) (35) and TRS API response

¹<https://github.com/pitagora-network/yevis-DAT2-cwl>

Table 2. Feature comparison with existing registries and Yevis-based registry. We focused on five characteristics of registries: acceptable workflows on each registry, workflow quality control responsibility, workflow testing responsibility, DOI assignment, and TRS compatibility.

Registry	URL	Acceptable workflows	Quality control by	Testing by	DOI	TRS
WorkflowHub	workflowhub.eu	No restrictions	Each developer	Each developer	No	Yes
Dockstore	dockstore.org	No restrictions	Each developer	Each developer	No	Yes
nf-core	nf-co.re	Generic workflows only	nf-core community	nf-core community	No	No
Built by Yevis	(a GitHub repo.)	Depend on administrator	Automated by Yevis	Forced by Yevis	Yes	Yes

(Figure 8) were updated through Yevis’s publication process to use URLs persisted by Zenodo. Requirements classified as “Validity” were addressed by running tests on GitHub Actions. The contents in the Yevis metadata file and the TRS response satisfy the validity requirements, such as workflow type, workflow language version, and the URL of the test results. Requirements classified as “Traceability” were addressed by describing, reviewing, and publishing them in the Yevis metadata file and TRS API response. From the above, we confirmed that Yevis successfully published the bacteria genome analysis workflow while addressing the defined requirements.

Discussion

Through our survey of existing workflow registries, such as Dockstore, WorkflowHub, and nf-core, it was revealed that they are maintained based on numerous contributions by various communities and the use of sufficient computer resources. While these established workflow registries accept submissions and are available for use by researchers, there are still cases in which there is a need to create a new workflow publication platform. For example, in the case of the Bioinformatics and DDBJ center, the institute (hereafter referred to simply as DDBJ) needed to have a collection of workflows that would be allowed to run on the WES on their computing platform. Therefore, we designed Yevis as a tool to help workflow developers create a registry to share their workflows. DDBJ used Yevis to create and then to maintain a workflow registry dedicated to workflows for use on the DDBJ WES (36).

Yevis can promote the concept of a distributed workflow registry model that underlies the specifications of the GA4GH Cloud Work Stream (18). In the distributed workflow registry model, researchers have the option to build their own workflow registry, rather than submitting to a centralized registry. The API standard for workflow registry specified by GA4GH enables a decentralized model, which promotes diversity in workflow development and in the research of analysis methods. Resource sharing, particularly of analysis methods, has a bigger impact on a community studying a minor target with limited human resources.

The Yevis system strongly relies on web services, such as GitHub and Zenodo. This is because we aimed to provide support to individuals or communities without sufficient computing resources, but this may result in a lock-in to these web services. Although we believe these services are reliable

enough to host valuable workflows, we designed the system to only use substitutable operations of those services, such as version management, file hosting, and continuous script execution. Yet we currently provide only the implementation that depends on those two services while it is technically possible to build a Yevis registry in an on-premise server. It is because the vendor-free system can only achieve part of our objectives to provide the effortless management of a registry. Here, we recognize a trade-off of building an academic tool using services run by commercial vendors, which requires further discussion in the communities. Automatic testing with GitHub Actions may also cause the issue of computational resource shortage. To extend the capability of testing, Yevis has the option to specify the location of an external WES endpoint to run the test, which also enables the testing with a specific computational request such as GPUs or job schedulers.

Compared to existing workflow registries that have a web form for workflow registration, the Yevis system provides only a command-line interface, Yevis-cli, as a method to submit a workflow. This is because we prefer to test workflows locally in advance of submission, while the existing registries test as part of a review process. By using the same test suite on both the submitter’s environment (local) and as part of the registry’s automatic process (remote), Yevis-cli ensures better reliability of the test results. This also helps to reduce the cost to a registry maintainer by ensuring a workflow is at least runnable on the submitter’s local environment.

The Yevis system provides a well-needed solution for research communities that aim to share their workflows and wish to establish their own registry as described. However, we recognize it still has some limitations. One of the challenges is the description of workflow testing. Writing tests for a workflow is difficult because the outputs may be heuristic values, may differ among tool versions, and the total amount of input and output data can be enormous. There is a need for a testing framework that packages test cases and results and generates assertions that allow for slight differences in the output. Debugging a workflow is also difficult because a typical workflow uses many tools that use various programming runtimes. Therefore, a framework is required to capture metrics of the test execution environment with these various runtimes.

Another challenge for the proposed distributed registry model is the findability of workflows. In the model where each developer is responsible for their content, the use of ap-

```

$ curl -fsSL https://pitagora-network.org/yevis-DAT2-cwl/tools/d03458d8-837c-4173-afa3-55ebe538b0b2/versions/1.0.0 | jq .
{
  "author": [
    "suecharo"
  ],
  "name": "DAT2-cwl - bacteria genome workflow",
  "url": "https://pitagora-network.github.io/yevis-DAT2-cwl/tools/d03458d8-837c-4173-afa3-55ebe538b0b2/versions/1.0.0",
  "id": "1.0.0",
  "descriptor_type": [
    "CWL"
  ],
  "verified": true,
  "verified_source": [
    "https://github.com/pitagora-network/yevis-DAT2-cwl/actions/runs/2317749577"
  ]
}

```

Fig. 8. TRS API response of the DAT2-cwl/bacteria-genome workflow. This JSON response is deployed on GitHub Pages by Yevis and is accessible via the HTTP protocol. The main workflow metadata in the TRS protocol is served at the path "/tools/{id}/versions/{version_id}". Two other possible paths for the associated files and the tests are "/tools/{id}/versions/{version_id}/files" and "/tools/{id}/versions/{version_id}/tests".

appropriate terms for describing workflow metadata can be an issue. A possible solution to improve the findability of workflows in distributed registries is to collect metadata in a centralized registry to curate them and create the search index. However, this will require a further challenge to distinguish the collected workflows using only metadata.

Many researchers agree that resource sharing is a key factor in the era of data science. As workflow systems and their communities grow, researchers have worked to share their data analysis procedures along with their data. Despite the fact that workflow systems are developed for automation, it sounds strange that maintaining workflow registries still relies on manual efforts. Through the development of Yevis, we found there are many possibilities for further automation in the process of resource sharing. Through the defined requirements for reusable workflows and a system that ensures them automatically, we believe that our work can contribute to moving open science forward.

Availability of source code and requirements

- Project name: Yevis-cli
 - Project home page: <https://github.com/ddbj/yevis-cli>
 - DOI: 10.5281/zenodo.6541109
 - biotoolsID: yevis-cli
 - Operating system(s): Platform independent
 - Programming language: Rust
 - Other requirements: Docker recommended
 - License: Apache License, Version 2.0
-
- Project name: Yevis-web
 - Project home page: <https://github.com/ddbj/yevis-web>
 - DOI: 10.5281/zenodo.6541031

- biotoolsID: yevis-web
- Operating system(s): Platform independent
- Programming language: TypeScript
- License: Apache License, Version 2.0

Availability of supporting data and materials

Data and materials related to the DAT2-cwl workflows described in the Section “Sharing workflows using Yevis” are available on GitHub and Zenodo as follows:

- GitHub repository for DAT2-cwl workflows (27)
- Workflow registry yevis-DAT2-cwl (37)
- Workflow browser for yevis-DAT2-cwl (38)

Declarations

List of abbreviations. API: Application Programming Interface; CI/CD: Continuous Integration/Continuous Delivery; CWL: Common Workflow Language; DDBJ: Bioinformatics and DDBJ Center; DNA: Deoxyribonucleic Acid; DOI: Digital Object Identifier; FAIR: Findable, Accessible, Interoperable, and Reusable; GA4GH: Global Alliance for Genomics and Health; HTTP: Hypertext Transfer Protocol; ID: Identifier; REST: Representational State Transfer; TRS: Tool Registry Service; UI: User Interface; URI: Uniform Resource Identifier; URL: Uniform Resource Locator; 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, the Life Science Database Integration Project, and the National Bioscience Database Center (NBDC) of the Japan Science and Technology Agency (JST). This study was also supported by the CREST program of the Japan Science and Technology Agency (Grant Number JP-MJCR17A1).

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.I., and T.O. 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 Elixir's BioHackathon Europe Series. We also would like to thank Ascade Inc. for their support with the software development.

References

- 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.
- 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.
- 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.
- Jeffrey M. Perkel. Workflow systems turn raw data into scientific knowledge. *Nature*, 573(7772):149–150, 2019. doi: 10.1038/d41586-019-02619-z.
- 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.
- Peter Amstutz, Maxim Mikheev, Michael R. Crusoe, Nebojša Tijanić, and Samuel Lampa. Existing workflow systems, 2021. <https://s.apache.org/existing-workflow-systems>.
- Laura Wratten, Andreas Wilim, and Jonathan Göke. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature Methods*, 18(10):1161–1168, 2021. doi: 10.1038/s41592-021-10254-9.
- Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, 46:W537–W544, 2018. doi: 10.1093/nar/gky379.
- Michael R. Crusoe, Sanne Abeln, Alexandru Iosup, Peter Amstutz, John Chilton, Nebojša Tijanić, Hervé Ménager, et al. Methods included: Standardizing computational reuse and portability with the common workflow language. *arXiv*, 2021. doi: 10.48550/arXiv.2105.07028.
- 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.
- Brian D. O'Connor, Denis Yuen, Vincent Chung, Andrew G. Duncan, Xiang Kun Liu, Janice Patricia, Benedict Paten, 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.

- 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.
- J. Koster and S. Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012. doi: 10.1093/bioinformatics/bts480.
- Felipe da Veiga Leprevost, Björn A. Grüning, Saulo Alves Afllitos, Hannes L. Röst, Julian Uszkoreit, Harald Barsnes, Marc Vaudel, et al. BioContainers: An open-source and community-driven framework for software standardization. *Bioinformatics*, 33(16):2580–2582, 2017. doi: 10.1093/bioinformatics/btx192.
- Carole Goble, Stian Soiland-Reyes, Finn Bacall, Stuart Owen, Alan Williams, Ignacio Eguinoa, Bert Driesbeke, et al. Implementing FAIR Digital Objects in the EOSC-Life Workflow Collaboratory. *Zenodo*, 2021. doi: 10.5281/zenodo.4605654.
- Philip A. Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilim, Maxime Ulysse Garcia, 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.
- Global Alliance for Genomics and Health. ga4gh/tool-registry-service-schemas, 2016. <https://github.com/ga4gh/tool-registry-service-schemas>.
- Heidi L. Rehm, Angela J. H. Page, Lindsay Smith, Jeremy B. Adams, Gil Alterovitz, Lawrence J. Babb, Maxmillian P. Barkley, et al. GA4GH: International policies and standards for data sharing across genomic research and healthcare. *Cell Genomics*, 1(2):100029, 2021. doi: 10.1016/j.xgen.2021.100029.
- Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R. Crusoe, Kristian Peters, and Daniel Schober. FAIR computational workflows. *Data Intelligence*, 2(1-2):108–121, 2020. doi: 10.1162/dint_a_00033.
- Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, 2016. doi: 10.1038/sdata.2016.18.
- Hirota Suetake and Tazro Ohta. Yevis: Getting Started, 2022. https://sapporo-wes.github.io/yevis-cli/getting_started. doi: 10.5281/zenodo.6793218.
- Global Alliance for Genomics and Health. ga4gh/workflow-execution-service-schemas, 2017. <https://github.com/ga4gh/workflow-execution-service-schemas>.
- Hirota 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.
- Dirk Merkel. Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014:2, 2014.
- Hirota Suetake and Tazro Ohta. ddbj/yevis-cli: 0.5.1 - actions_example/yevis-test-pr.yml, 2022. https://github.com/ddbj/yevis-cli/blob/0.5.1/actions_example/yevis-test-pr.yml. doi: 10.5281/zenodo.6793218.
- Hirota Suetake and Tazro Ohta. ddbj/yevis-cli: 0.5.1 - actions_example/yevis-publish-pr.yml, 2022. https://github.com/ddbj/yevis-cli/blob/0.5.1/actions_example/yevis-publish-pr.yml. doi: 10.5281/zenodo.6793218.
- Pitagora Network members. pitagora-network/DAT2-cwl: 1.1.1, May 2022. doi: 10.5281/zenodo.6565977.
- Bono Hidemasa and Atsushi Shimizu. *Next Generation Sequencer DRY Analysis Manual 2nd Edition*. Gakken Medical Shujunsha, 2019.
- Pitagora Network members. GitHub - pitagora-network/DAT2-cwl: 1.1.1 - workflow/bacteria-genome, May 2022. doi: 10.5281/zenodo.6565977.
- Wei Shen, Shuai Le, Yan Li, and Fuquan Hu. SeqKit: A cross-platform and ultrafast toolkit for FASTA/Q file manipulation. *PLOS One*, 11(10):e0163962, 2016. doi: 10.1371/journal.pone.0163962.
- Simon Andrews. FastQC: A quality control tool for high throughput sequence data, 2010.
- Shifu Chen, Yanqing Zhou, Yaru Chen, and Jia Gu. fastp: An ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics*, 34(17):i884–i890, 2018. doi: 10.1093/bioinformatics/bty560.
- Rei Kajitani, Kouta Tshimoto, Hideki Noguchi, Atsushi Toyoda, Yoshitoshi Ogura, Miki Okuno, Mitsuru Yabana, et al. Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads. *Genome Research*, 24(8):1384–1395, 2014. doi: 10.1101/gr.170720.113.
- Hirota Suetake. DAT2-cwl/bacteria-genome workflow files uploaded to Zenodo by Yevis, May 2022. doi: 10.5281/zenodo.6545122.
- Hirota Suetake. Yevis metadata file for the DAT2-cwl/bacteria-genome workflow, May 2022. doi: 10.5281/zenodo.6572565.
- Hirota Suetake and Tazro Ohta. ddbj/workflow-registry: 1.0.2, 2022. doi: 10.5281/zenodo.6719845.
- Hirota Suetake. pitagora-network/yevis-DAT2-cwl: 1.0.0, 2022. doi: 10.5281/zenodo.6572565.
- Hirota Suetake. pitagora-network/yevis-DAT2-cwl-browser: 1.0.0, 2022. doi: 10.5281/zenodo.6575089.

Point-by-point response to reviewers' comments

GIGA-D-22-00187 Workflow sharing with automated metadata validation and test execution to improve the reusability of published workflows Hirota Suetake; Tsukasa Fukusato; Takeo Igarashi; Tazro Ohta GigaScience

We would like to thank reviewers for their thoughtful comments and constructive feedbacks. We have added a summary figure and some additional paragraphs to make things clear, and made changes in the figures from the previous version of the manuscript.

Our responses and explanations to the issues pointed by each reviewer are follows.

Reviewer #1

1. Based on your survey of existing systems, could you possibly make a figure or table that showcases the features supported/not supported by these different systems, including yours?

Response: Thank you for the suggestion that can highlight the contribution of our work. We added the Table 2 in the Result section to compare the characteristics of WorkflowHub, Dockstore, nf-core, and a yevis-based registry. We compared them with three aspects; diversity, reliability, and usability of registered workflow.

2. Thoughts on security/cost safeguards? Perhaps beyond the scope, but it does seem like a governing group needs to define some limits to the testing resources and be able to enforce them. If I am a bad actor and programmatically open up 1000 PRs of expensive jobs, I'm not sure what would happen. Actions and artifact storage aren't necessarily free after some limit.

Response: We first had security concerns with the functionality in GitHub. Then, to solve problems of offensive actions by using malicious users via Pull Requests, we use GitHub's first-time contributor restriction. In this system, users (who first contribute to the repository) cannot perform a GitHub action without permission from the administrator. That is, the owners can easily reject "Pull Requests" from suspicious users, i.e., avoiding possible security issues.

As for the resource limitation of GitHub Actions, we solve the problem by using an external WES. In addition to a lack of resources, GitHub Actions may not be sufficient for testing workflows that only work on specific system configurations, such as GPUs or job schedulers. In such cases, the registry owner can check the testing portability by running the test on a remote WES instance. We added sentences to the Discussion section about the limitations of GitHub resources, the incident possibilities, and solutions to prevent them.

We added the following sentences to the Discussion section:

Automatic testing with GitHub actions may also cause the issue of computational resource shortage. To extend the capability of testing, Yevis has the option to specify the location of an external WES endpoint to run the test, which also enables the testing with a specific computational request such as GPUs or job schedulers.

3. What is the flow for simply updating to a new version of an existing workflow? (perhaps this could be in your docs, not necessarily this manuscript).

Response: We appreciate the comment about the version updates, which is one of the critical features of our implementation. The system identifies a workflow by its remote URL of the primary workflow in the metadata. Therefore, if the user modifies the metadata without changing the URL and bumps up the version information, the system recognizes that they are different versions of the same workflow. Users can use different versions of the workflow by specifying the TRS endpoint with a path including version information, such as "tools/0d2ae4c2-fe4c-48f7-811a-ac277776533e/versions/1.0.0".

Users can upgrade the registered workflow by performing the following steps.

1. Edit the existing yevis-metadata.yml file.
2. Bump the workflow version in the metadata.
3. Run tests using yevis-cli and send a PR to the repository.
4. Follow the same steps as adding a new workflow, and the new version will be published.

We added the above instruction to the online documentation.

4. CWL is an example of a workflow language that developers can extend to create custom "hints" or "requirements". For example, seven bridges does this in cavatica where a user can define aws spot instance configs etc. WDL has properties to config GCP images. It seems like in these cases, tests should only be defined to work when running "locally" (not with some scheduler/specific cloud env). But the author's do mention that tests will first run locally on the user's environment, so that does kind of get around this.

Response: As we see the importance of the testing portability, we implemented the system with a strict specification. Developers can push the workflows that passed the local test to the registry, and the owners can merge those that passed the remote test. However, as the reviewer points out, specific compute-dependent tests exist in actual use cases and cannot be ignored. Allowing tests to be run outside the developer's own environment can solve the testing portability problem, i.e. the testing on a computing environment that anyone can access, even if not on GitHub. Therefore, the ability of our GitHub action to request tests to external WES instances can solve the problem. However, another problem is preparing a WES instance that can correctly run the tests for that workflow. We believe it is the responsibility of developers who want to share their workflows to prepare an environment where anyone can execute the tests correctly. Perhaps the preparation of a testing environment is something that the community should discuss.

5. For the "findable" part of FAIR, how possible is it to have "tags" of sort associated with a wf record so things can be more findable? I imagine when there is a large repository of many workflows, being able to easily narrow down to the specific domain interest you have could be helpful.

Response: It is crucial to curate the metadata in the registry with tags or ontology terms that increase findability. The findability issue is a downside of the de-centralized registry model. Although it is possible to add tags as part of the TRS response specification, tagging with uncontrolled vocabulary by each workflow developer will not improve the findability. We believe that a centralized search index is in need to distinguish workflows from other similar or similar but different workflows available online. Therefore, a possible strategy is to register the metadata in a centralized registry such as WorkflowHub, while the workflow body is hosted and test runs on GitHub/Zenodo, as we proposed.

We have added this point as a weakness of the de-centralized model in the Discussion as following:

Another challenge for the proposed distributed registry model is the findability of workflows. In the model where each developer is responsible for their content, the use of appropriate terms for describing workflow metadata can be an issue. A possible solution to improve the findability of workflows in distributed registries is to collect metadata in a centralized registry to curate them and create the search index. However, this will require a further challenge to distinguish the collected workflows using only metadata.

Reviewer #2

Main concern

I have one major gripe though, blocking acceptance: The choice to only support GitHub for hosting. There is a growing problem in the research world that more and more research is being dependent on the single commercial actor GitHub, for seemingly no other reason than convenience. Although GitHub to date can be said to have been a somewhat trustworthy player, there is no guarantee for the future, and ultimately this leaves a lot of research in an unhealthy dependence on this single platform. As a small note of a recent change, is the proposed removal of the promise to not track its users (see <https://github.com/github/site-policy/pull/582>). As such a central infrastructure component for research as a workflow registry has an enormous responsibility here, as it may greatly influence the choices of researchers in the future to come, because of encouragement of what is "easier" or more convenient to do with the tools and infrastructure available. With this in mind, I find it unacceptable for a workflow registry supporting open science and open source work to only support one commercial provider. The authors mention that technically they are able to support any vendor, and also on-premise setups, which sounds excellent. I ask the authors to kindly implement this functionality. Especially the ability to run on-premises registries is key to encourage research to stay free and independent from commercial concerns.

Response: We understand the reviewer's concern about the lock-in risk to a commercial service well. It is not our desire to be locked into a specific service, and we are aware of the risk that these services (and every other service run by a vendor) may become unavailable to everyone. However, at the same time, it is impractical for us to implement and provide the "perfect lock-free" version of the system with production-level quality as well. We would like to explain the reasons below.

In this study, we aimed to achieve two major objectives: the independency from computational resources maintained by individual developers, and the automation to reduce the developers' tasks that make workflows reusable.

For the first objective, the academic communities know from their experience how difficult it is to maintain one's server for decades, at least for most of the developers in research. We think the main reason why many researchers rely on GitHub is simple, they do not need to maintain their git servers. This is also why we chose to use services run by vendors while realizing the risk of lock-in.

For the second objective, we had to retrieve the metadata of workflows automatically. Therefore, we used the GitHub API as a core component, which is not straightforward to replace with a plain on-premise git server.

Therefore, we have to give up these two strong benefits if we wanted to provide the same functionalities with a possible on-premise Yevis implementation. We still can think of the lock-free version of the registry, however, it will look like a completely different one as follows:

1. Workflow developers manually create metadata.yml and run the tests by hand using WES.
2. Developers submit the metadata to the registry administrator using a communication channel such as a mailing list.
3. The administrator runs tests using WES based on the received metadata.yml and reviews them.
4. Upload the reviewed workflow to any file server and publish it.

With these steps, partially supported by the current yevis-cli, one can have a workflow registry completely vendor lock-free. However, it is not as far efficient as the system using GitHub and Zenodo.

We believe our contribution is not the system itself but rather the distributed registry model in which communities can have their platforms by automating the quality assurance tasks. We hope that GitHub will not become evil, but if it does, we will re-implement our system with other services.

To clarify the benefit and the risk of vendor lock-in, and to explain the form of the system without those services, we added the sentences to the discussion section as follows:

Yet we currently provide only the implementation that depends on those two services while it is technically possible to build a Yevis registry in an on-premise server. It is because the vendor-free system can only achieve part of our objectives to provide the effortless management of a registry. Here, we recognize a trade-off of building an academic tool using services run by commercial vendors, which requires further discussion in the communities.

Minor concerns

1. I think the manuscript is a missing citation to this key workflow review, as a recent overview of the bioinformatics workflows field, for example together with the current citation [6] in the manuscript: Wratten, L., Wilm, A., & Göke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nature methods, 18(10), 1161-1168. <https://www.nature.com/articles/s41592-021-01254-9>

Response: Thank you for pointing out the important citation. We added the citation as suggested.

2. Although it might not have been the intention of the authors, the following sentence sounds unnecessarily subjective and appraising, without data to back this up (rather this would be something for the users to evaluate): The Yevis system is a great solution for research communities that aim to share their workflows and wish to establish their own registry as described. I would rather expect wording similar to: "The Yevis system provides a [well-needed] solution for ..." ... which I think might have been closer to what the authors intended as well.

Response: We thank the precise suggestion to the statement. We edited the sentence as suggested.

Reviewer #3

One of my main concerns is that this work presents a technical implementation but lacks an architecture diagram or a big picture that would clarify the contributions of this work with respect to the features provided by the external platforms (GitHub, Zenodo). This would also help the reader understand how the solution proposed can be reused with possibly other services.

Response: We agree with the reviewer's concern that the system's overall architecture is not clear to the readers. Although we expected that the Figure 1 in the original submission shows the overall procedure with the components we used, it focuses on the steps of the submission, not the environment where the system is running. We added a new diagram as Figure 1 to indicate the components where each process is running and how they are connected each other.

It is also hard to understand the technical solutions when target users and their typical needs have not been clearly stated beforehand. It seems to be very complicated for non-developers to use or operate Yevis, especially when testing workflows through the "GitHub actions" infrastructure. Is the Yevis platform only targeting workflow developers? It was not easy to understand the benefits offered to research communities aimed at sharing/reusing workflows.

Response: Firstly, we would like to thank the reviewer for pointing out that it is not clear about the system's target user is. We agree that it was implicit in the manuscript that the technical level of the users expected to use the system. We expect workflow developers as the target users of the Yevis system, i.e., developers and researchers with the skills to design and implement arbitrary workflows using the workflow language. However, the procedure for publishing workflows using the Yevis system does not require any of the difficulties noted by the reviewer. The repository template used in the Yevis setup procedure already has the GitHub action for workflow testing that the system automatically executes when a pull request is received. Thus, the users do not need to write any GitHub actions but prepare the test data for their workflows and write their metadata. As indicated in the new system overview figure, many procedures exist to publish. However, the system automates them except for the writing of metadata and the administrator's review. We believe that this labor-saving automation is the contribution of this research.

To clarify the target users, we added a sentence to the Background section as follows:

The system expects developers and researchers who design workflows using workflow languages as users, although it does not require advanced computer skills to operate the system.

Regarding the background section, GA4GH-TRS is not introduced while mentioned as part of the results. It's hard for the reader to understand how the use of this standard contributes to better workflow sharing (metadata?) or better reuse (tests?).

Response: We added the sentence to the second paragraph of the System design section as follows:

The TRS defines the specification of computational tool/workflow metadata representation, including workflow's URI, used language, version, etc. It ensures the interoperabilities among different tool/workflow registries and enables workflow engines to retrieve the information to execute a workflow maintained at a remote server, which improves reusability of published workflows.

Other workflow registries such as WorkflowHub or NF-Core have been described in the background section but a dedicated state-of-the-art section would have allowed the authors to provide more details on the positioning of Yevis. Some related works should also be part of the analysis such as BIAFLOWS also providing a benchmarking environment, or OpenEBench.

Response: We added the Table 2 to compare the existing registries, including the WorkflowHub, nf-core, and Dockstore, to compare their characteristics with a Yevis-based registry. We reviewed the BIAFLOWS and OpenEBench and we recognized their features of tool/workflow collections and evaluations. However, their objectives are to provide a summary to compare different computational tools for better tool selection, which is a slightly different scope from the aim of this study to publish workflow with testing results for

better trust. Therefore, we exclude them from the table, but we thank the reviewer for pointing out the relevance of our system with those excellent community resources.

The live deployment URL of the Yevis system should be provided in the paper so that readers can browse/reuse already registered workflow. The link provided in the source code repository only shows 4 workflows and not the DAT2 workflow used in the "proof of concept" section. Is the system limited to some workflow engines ? Would it be possible to register and test Galaxy workflows for instance ?

Response: In the original manuscript, we mentioned the registry URL of the DAT2-cwl registry in the reference section (37), but we recognized it was not clear to the readers. We added the URL to the DAT2-cwl registry (<https://github.com/pitagora-network/yevis-DAT2-cwl>) to the "sharing workflows using Yevis" section and the repository of yevis-cli. The registry with four workflows is the DDBJ workflow registry, which is different from the one we have shown as a demo in the manuscript. We also have corrected the resources mentioned in the manuscript.

The yevis system itself does not have a limitation on workflow engines but requires a WES endpoint to test the given workflow. Thus, if we had a WES instance that consumes a Galaxy workflow the system can accept a Galaxy workflow. Currently, we have a WES instance available for Common Workflow Language, Nextflow, Workflow Description Language, and Snakemake.

Regarding tests of workflows, the files associated to HiSAT2 on the Pitagora workflow were not accessible (404 not found). I also had some difficulties when trying to inspect the results of the automated tests in the GitHub actions. For this workflow, [Add workflow: Pitagora CWL - Download SRA · ddbj/workflow-registry@89961a4 · GitHub](#), the logs were expired and thus no more accessible. This highlights the challenge of relying on external computing services to run possibly long and costly executions, even with test data.

Response: We thank the reviewer for pointing out a critical issue of the current implementation. We missed this end-of-life of the GitHub actions artifact during our development phase. This GitHub action artifact is for two objectives, and we expect this limitation does not affect the current Yevis procedure. First, the artifact is supposed for the review by the registry administrator. If 90 days have passed since the automatic execution and the test log is missing, the admin can re-run the action and review the artifact again. Second, the artifact is also valuable as a workflow run provenance. For this purpose, we have added a function in the WES instance to generate ResearchObject-crate (RO-crate), a standard specification to record the research artifact, to let the Yevis system store the test log in the Zenodo repository.

For long and costly workflow execution problem, the system has an option to run tests with the external WES endpoint shown in the new summary figure. It brings the availability issue of the testing instance but can solve the problem of the resource limitation of the GitHub actions.

Regarding the validation of metadata, very few informations are provided. Are all metadata fields mandatory ? are some fields recommended ? Which kind of validation is performed ? How the result of validation is returned to users ? Regarding the metadata themselves, how do they comply with community emerging standards such as Bioschemas or RO-crate ? At the time of the review, it was not possible to find any semantic annotations in the Yevis web page, thus limiting the discoverability and the interoperability of workflows descriptions.

Response: For each field in the metadata, the system performs a type check, license check, URL validation, remote file existence, etc. We added the information on these validation items to the CLI's GitHub repository. (<https://github.com/sapporo-wes/yevis-cli#validate>) The CLI checks these metadata items written by the user, and if it finds an invalid field, the CLI outputs an error message and prompts the user to correct it. As the reviewer points out, our specification for workflow metadata overlaps with some of the community-developed standards. Currently, for workflow test run logs, we are working with the community to apply the workflow run crate under development by the RO-crate community. However, as we prioritize the readability and editability of Yevis metadata, the approach could be to adopt a proprietary format and then convert it to a standard by the community.

Because of technical limitations in the current implementation, embedding tags in the pages is not practical to improve findability. As we implemented the Yevis-web in the JavaScript Single Page Application framework, it is not possible to embed different annotation tags in the HTML source of each workflow page.

However, we believe workflow findability should not be left to individual workflow developers but should be a centralized effort by a central registry. Therefore, we would like to enhance the findability of workflows by having individual communities manage their workflows with Yevis while a centralized registry such as WorkflowHub collects these metadata.

Finally, the discussion and future works sections could be enriched to address for instance.

- the scalability of the approach with possibly long or costly tasks when testing workflows
- the interoperability of this platform with other registries
- the genericity of the approach (is it applicable in the context of other scientific disciplines)
- the use of this platform to compare or benchmark workflow executions based on predefined test datasets

Response: We would like to thank the reviewer's suggestion to make the Discussion section more constructive. We added the following topics in the Discussion section to reflect the reviewers' feedback.

- Using external WES endpoint to overcome the resource limitation of GitHub actions: also brings the concern of availability (3rd paragraph)
- The automation of quality control procedure using CI/CD of the public services: its limitation and possibility (3rd paragraph)
- The interoperability of the Yevis registry and possibilities of coexistence with the other centralized registries for findability (6th paragraph)

We excluded the topic of benchmarking because we think it can confuse the readers as if the Yevis system can use for that purpose. We expect benchmarking feature should be provided on a different layer than that of the registry.

Minor comments

Figures 2 and 7 seems to be very similar. Only one should be kept.

Response: We agree with the reviewer's assessment. To avoid the confusion, we removed the Figure 7 which showed the metadata file modified by the yevis system. We remained Figure 2 and added the explanation that the metadata will be modified.

How are test specified, is the specification generic enough? How does it support multiple workflow engines?

Response: Yevis's specification of the test run conforms to the GA4GH WES specification. In yevis-cli, the option `--wes-location` specifies the WES instance on which the system runs the test. As WES implementations usually have a single workflow engine, users need to use a WES capable of executing the workflow's language. As an exception, Sapporo, one of the WES implementations we have developed, provides users the option to specify a workflow engine from several engines covering CWL, WDL, Nextflow, and Snakemake. The yevis-cli command uses Sapporo via Docker if users have no WES preference.

We added sentences to emphasize the option to specify an external WES for testing instance as follows:

Automatic testing with GitHub Actions may also cause the issue of computational resource shortage. To extend the capability of testing, Yevis has the option to specify the location of an external WES endpoint to run the test, which also enables the testing with a specific computational request such as GPUs or job schedulers.

The paragraph on decentralization in the discussion is confusing. All workflow executions seem to be centralized on the GitHub infrastructure with no control on data or computation placement. The same happens for data on the Zenodo infrastructure.

Response: Thank you for your comments. We think there is a misunderstanding about the term decentralization. In the submission stage, we described "Yevis can promote the concept of a distributed workflow registry model that underlies the specifications of the GA4GH Cloud Work Stream," so this paragraph claim is the idea of decentralization of workflow registries, not computation placement. Since we emphasize that the communities have an option to build their own registry, not relying on a centralized registry such as WorkflowHub or nf-core, we added a sentence to clarify the idea as follows:

In the distributed workflow registry model, researchers have the option to build their own workflow registry, rather than submitting to a centralized registry.

As mentioned above, the computation also can be decentralized using the external WES instances.

DAT2-cwl is not registered in the live deployment of Yevis

Response: The original manuscript has referred external GitHub repositories and the resources, which may confuse readers. We added the URL in the sentence along with the citation to lead the readers to the appropriate resource.