nature methods
PERSPECTIVE
https://doi.org/10.1038/s41592-021-01254-9

Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers

Laura Wratten[1], Andreas Wilm[2] and Jonathan Göke[1]✉

The rapid growth of high-throughput technologies has transformed biomedical research. With the increasing amount and complexity of data, scalability and reproducibility have become essential not just for experiments, but also for computational analysis. However, transforming data into information involves running a large number of tools, optimizing parameters, and integrating dynamically changing reference data. Workflow managers were developed in response to such challenges. They simplify pipeline development, optimize resource usage, handle software installation and versions, and run on different compute platforms, enabling workflow portability and sharing. In this Perspective, we highlight key features of workflow managers, compare commonly used approaches for bioinformatics workflows, and provide a guide for computational and noncomputational users. We outline community-curated pipeline initiatives that enable novice and experienced users to perform complex, best-practice analyses without having to manually assemble workflows. In sum, we illustrate how workflow managers contribute to making computational analysis in biomedical research shareable, scalable, and reproducible.

Increased throughput, new technologies, and higher sample numbers have contributed to the production of massive amounts of biological data[1,2]. While some of the largest data generators are national and international consortia, even research by individual teams now frequently uses high-throughput technology and sizable bioinformatics analysis. The analysis of biological data is driven by the development of an extensive array of open-source software tools[3,4]. Most of these tools carry out a single specialized step, which when chained together enables the creation of complex analysis workflows that process and analyze the increasing amount of biological data. However, with complex chains of steps, variability in operating systems and computational resources, and ambiguities with tool versioning and documentation[5–7], reproducibility of analysis workflows has become a key issue in computational biology[8,9].

In order to minimize the number of manual steps that are required to execute an analysis workflow, computational pipelines automatically chain together multiple tools (Fig. 1a). Historically, computational pipelines were developed using custom scripts or Make files[10,11] (Fig. 1b). These traditional pipelines greatly simplify the recurrent analysis of data. However, traditional pipelines are usually highly coupled to their local compute infrastructure; cannot resume a failed run; lack sufficient documentation, parameter tracking, and tool versioning; and require manual installation when running on another device, making them difficult to share and maintain, and making produced results often impossible to reproduce[11–13]. Even when a user of a traditional pipeline installs relevant software and dependencies, as well as obtaining the exact software versions for each tool, analysis results can still differ[5,6].

Data-heavy fields, such as banking, the automotive industry, and technology start-ups, have successfully used workflow managers to handle complex data analytics workflows (for example, https://www.pachyderm.com/use-cases/, https://github.com/spotify/luigi, https://airbnb.io/projects/airflow/, https://github.com/Netflix/metaflow, https://github.com/uber/cadence). Workflow managers provide a framework for the creation, execution, and monitoring of pipelines. In recent years, a number of workflow managers have been specifically designed for biomedical data[11,14], directly addressing the need of computational analysis workflows in research and health care. Bioinformatics workflow managers offer integration with containers, package managers, and cloud computing, while providing automatic resource management[15]. Implementing a pipeline with a workflow manager can simplify pipeline development, maintenance, and use, while enabling portability and improving reproducibility (Fig. 1c).

Workflow managers provide a powerful tool for pipeline development, yet many different frameworks exist that differ in their ease of use, ability for customization, documentation, and requirements of prior programming knowledge[11]. Here, we introduce the advantages of workflow managers compared with traditional pipelines and compare some of the existing approaches. We review pipeline repositories that provide curated collections of pipelines to avoid re-implementing best-practice analysis workflows. We aim to provide a guide and overview to facilitate the use of workflow managers for computational and noncomputational users, while highlighting the concepts that we believe will become essential for data-driven research and applications in high-throughput biology.

Data provenance

In computational biology, one of the major challenges to enabling reproducibility is that any change in software versions, parameters, or reference annotation versions can alter the results[16–20]. Data provenance describes this trail of methods, versions, and arguments that were used to generate a set of files[21].

Workflow managers automate the process of input parameter and software tool version tracking for computational pipelines. They provide the option to generate execution reports with detailed information, such as input parameters to the pipeline; the execution environment; the software version of the workflow manager and tools used; resource usage information, including execution time and CPU usage; and parameters for each individual tool and a visualization of the pipeline steps. While execution reports are file-specific, the workflow itself can be publicly archived and made citable by obtaining a version-specific digital object identifier (DOI) through Zenodo[22,23]. This provides a high level of documentation on

[1]Genome Institute of Singapore, Singapore, Singapore. [2]ImmunoScape, Singapore, Singapore. ✉e-mail: gokej@gis.a-star.edu.sg

Table 1 | Overview of workflow managers for bioinformatics (top, editable version; bottom, image version)

Tool	Class	Ease of use[a]	Expressiveness[b]	Portability[c]	Scalability[d]	Learning resources[e]	Pipeline initiatives[f]
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
KNIME	Graphical	●●●	●○○	○○○	●●◐	●●●	●●○
Nextflow	DSL	●●○	●●●	●●●	●●●	●●●	●●●
Snakemake	DSL	●●○	●●●	●●◐	●●●	●●○	●●●
GenPipes	DSL	●●○	●●●	●●○	●●○	●●○	●●○
bPipe	DSL	●●○	●●●	●●○	●●◐	●●○	●○○
Pachyderm	DSL	●●○	●●●	●○○	●●○	●●●	○○○
SciPipe	Library	●●○	●●●	○○○	○○○	●●○	○○○
Luigi	Library	●●○	●●●	●○○	●●◐	●●○	○○○
Cromwell + WDL	Execution + workflow specification	●○○	●●○	●●●	●●◐	●●○	●●○
cwltool + CWL	Execution + workflow specification	●○○	●●○	●●◐	○○○	●●●	●●○
Toil + CWL/WDL/Python	Execution + workflow specification	●○○	●●●	●◐○	●●●	●●○	●●○

Please refer to Supplementary Table 1 for details. This information is based on online documentation and manuscripts and may not be reflective of the current state of the projects. Scores for Galaxy are based on the graphical user interface. [a]Ease of use: graphical interface with execution environment (score of 3), programming interface with in-built execution environment (score of 2), separated development and execution environment (score of 1). [b]Expressiveness: based on an existing programming language (3) or a new language or restricted vocabulary (2), primary interaction with graphical user interface (1). [c]Portability: integration with three or more container and package manager platforms (3), two platforms are supported (2), one platform is supported (1). [d]Scalability: considers cloud support, scheduler and orchestration tool integration, and executor support. Please refer to Supplementary Table 1. [e]Learning resources: official tutorials, forums, and events (3), tutorials and forums (2), tutorials or forums (1). [f]Pipelines initiatives: community and curated (3), community or curated (2), not community or curated (1).

how files are processed, enabling transparency, code sharing, and long-term reproducibility through data provenance.

Portability

Pipeline reports and DOIs ensure that a workflow can be run with identical parameters and software versions. However, executing the same pipeline code on another machine or operating system can still be impossible, for example, owing to missing dependencies or incompatible software versions. In contrast, a workflow that is portable can be executed with the same functionality across different platforms (provided that minimum hardware requirements, such as CPU and memory, are met for the workflow).

Workflow managers utilize two technologies to automate the software installation process and ensure portability across different platforms: package managers and containerization software[9].

Package managers automate the process of installing and configuring software, enabling a user to obtain all tools and dependencies required to execute a pipeline with a single command, eliminating the need to locate and manually install tools with distinct installation requirements[24,25]. Examples of package managers are Homebrew (https://brew.sh/) for MacOS and Linux, and Conda (https://conda.io), which provides an isolated environment for pipeline execution in addition to efficiently installing pipeline dependencies[26]. In particular, Bioconda (https://bioconda.github.io/), a Conda channel specializing in bioinformatics, has contributed to the availability and ease of installation of bioinformatics tools. Bioconda provides over 8,000 maintained and curated Conda recipes for bioinformatics software[27]. Package managers enable platform-independent software installation, largely eliminating the need for pipeline developers to write their own platform-specific recipes for dependency installation.

Containers, on the other hand, are a lightweight, configurable virtualization technology that allows packaging and distribution of pipelines and their corresponding dependencies in a self-contained and platform-independent manner[28]. Commonly used software for containerization in bioinformatics includes Docker (https://www.docker.com/) and Singularity[29]. The repository Dockstore aggregates containerized bioinformatics tools and workflows into a searchable repository[30], while the BioContainers[31] community initiative provides prebuilt containers to be used with Docker and Singularity. BioContainers was specifically designed to host bioinformatics software, transforming the way bioinformatics tools are installed and used, as well as making containerized software accessible to noncomputational researchers.

Workflow managers seamlessly integrate containerization software and package managers, enabling users to install dependencies and run a pipeline without the requirement for preinstalled local versions. Some workflow managers enable this option with a single command-line flag, thus ensuring cross-platform portability for scientific workflows. This approach to portability also increases reproducibility across different compute platforms, as factors such as the choice of operating system have been shown to influence analysis results[12,32].

Scalability

The rapid rise of high-throughput technologies has greatly increased the scale and complexity of data that are routinely generated and analyzed. Being able to run the analysis of biological data at any scale in a fast and resource- and cost-efficient way has therefore become a key requirement for computational pipelines[33,34]. Scalability of pipelines encompasses two aspects: efficient resource management and the ability to handle any size and quantity of input data[35,36].

Resource management is particularly important for bioinformatics workflows that consist of multiple steps that each have different CPU and memory requirements. Most workflow managers efficiently utilize resources through parallelization of the different steps[37]. Workflow managers implement parallelization in a number of ways, including static scheduling, job-queue scheduling, and adaptive scheduling[38]. The choice of implementation of workflow scheduling as well as whether parallelization occurs on the data, task, or pipeline level influences both workflow performance and the way users write workflows for the system[39].

Table 2 | Overview of bioinformatics pipeline projects

Pipeline initiative	Tool	Curated[a]	Community[b]	Citable[c]	Pipelines[d]
nf-core	Nextflow	✓	✓	✓	27
snakePipes	Snakemake	✓	✗	✓	9
Snakemake-Workflows	Snakemake	✓	✓	✓	7
GenPipes	GenPipes	✓	✗	✓	12
Galaxy Community	Galaxy	✗	✓	✗	>1,000
BioWDL	WDL	✓	✗	✓	17
WARP	WDL	✓	✗	✗	8
KNIME Hub	KNIME	✗	✓	✗	>1,000

[a]Curated: peer review or best practice, ability to ask questions of developers, and extensive documentation with clear contribution guidelines and testing. [b]Community: not hosted on institute-specific infrastructure and developers from various institutes. [c]Citable: provides guidelines for pipeline citation. [d]Pipelines: number of released pipelines (not counting drafts or prereleases).

By dynamically scheduling independent tasks during execution of the pipeline, unused resources can be utilized without affecting the most resource-intensive steps. This process enables the effective handling of large datasets and minimizes bottlenecks that increase running time. Workflow managers provide a high degree of flexibility to control resource utilization, as memory and compute requirements can be specified for each step or for the entire workflow[40].

Even with optimal resource management, the maximum scalability of traditional pipelines is limited by the local compute infrastructure. Workflow managers achieve scalability beyond the local infrastructure by providing in-built support for high-performance computing environments and cloud computing services[41–44]. Workflow managers often provide simple options to execute the same pipeline on different compute infrastructures, with direct support for major cloud compute providers and popular scheduling software[45]. In addition, some workflow managers can use container orchestration systems, such as Kubernetes (https://kubernetes.io/) and Docker Swarm (https://docs.docker.com/engine/swarm/), to automatically manage the scheduling and deployment of containers, further enabling the effective utilization of available resources.

The optimized resource management and support for major execution infrastructures ensure that pipelines written with workflow managers can be scaled for the efficient analysis of small and large datasets.

Re-entrancy

With the rise of pay-as-you-use cloud computing and the use of pipelines for clinical and industrial applications, cost and time-to-result have become important factors in pipeline execution[42,46,47]. Computational workflows often have a large number of steps that can be resource intensive. When the execution of a traditional pipeline is disrupted owing to errors or manual intervention, it has to be restarted from the first step, thereby recomputing already computed results and thus wasting computational resources.

Workflow managers can handle such events by enabling re-entrancy. Re-entrancy allows users to run a pipeline from its last successfully executed step, rather than from the beginning, in the case of a disruption. Re-entrancy also minimizes the need for recalculation of frequent data-processing steps such as saving reference genomes and index creation. To achieve this, workflow managers use caching to save intermediate results and data files and compare this with the expected output to generate only necessary files[48]. Workflow managers vary in their implementation of re-entrancy, which may result in differences in storage overheads because of intermediate file generation. Re-entrancy saves significant time and compute resources and is a key advantage of workflow managers[49].

A practical guide to workflow management tools for pipeline development

There are over 150 workflow managers currently in use and under development (https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems; https://github.com/pditommaso/awesome-pipeline).

One of the main distinctions between workflow managers is the trade-off between ease of use, flexibility, and feature richness. While some workflow managers enable the implementation of pipelines with simple graphical user interfaces, others require minimal or more advanced programming knowledge. The increased level of expressiveness and abstraction enables the implementation of more flexible and powerful pipelines, often required by bioinformatics core units or collaborative consortia. Additional criteria that distinguish workflow managers are the availability of learning resources, access to preimplemented pipelines, and differences in portability and scalability (Table 1 and Supplementary Table 1). While we believe that a comparison of workflow managers can provide a useful overview, criteria such as ease of use and expressiveness are based on definitions that capture only one aspect and as such are subjective and will differ for each user. Many more workflow managers exist, and it is important to note that this is a simplified representation that does not capture most of the aspects and features that are often dynamically evolving and as such should not be interpreted as a ranking. Criteria such as personal preferences of programming languages, access to a local community or support within an institute, uptake by the global community, long-term support, and active development can often be equally or even more important to consider.

Graphical workflow managers: point-and-click pipeline development

Graphical workflow managers support user interaction through a graphical user interface. They provide a point-and-click interface for users to drag and drop tools into workflows and chain them together, enabling the creation of complex computational pipelines without programming experience.

Examples of graphical workflow managers are Galaxy, a web-based platform for bioinformatics workflows[50], and KNIME, a graphical tool for building machine-learning and data science workflows[51,52]. Galaxy features over 8,000 tools (https://toolshed.g2.bx.psu.edu/)[53] and an abundance of learning resources (https://galaxyproject.org/learn/; https://training.galaxyproject.org/)[54]. Galaxy is actively maintained by a core team and an active community, with a large number of pipelines being published[55]. The Galaxy project provides public infrastructure to run workflows without incurring costs as well as functionality for specialized data analysis and visualization that go beyond the functionality of most other

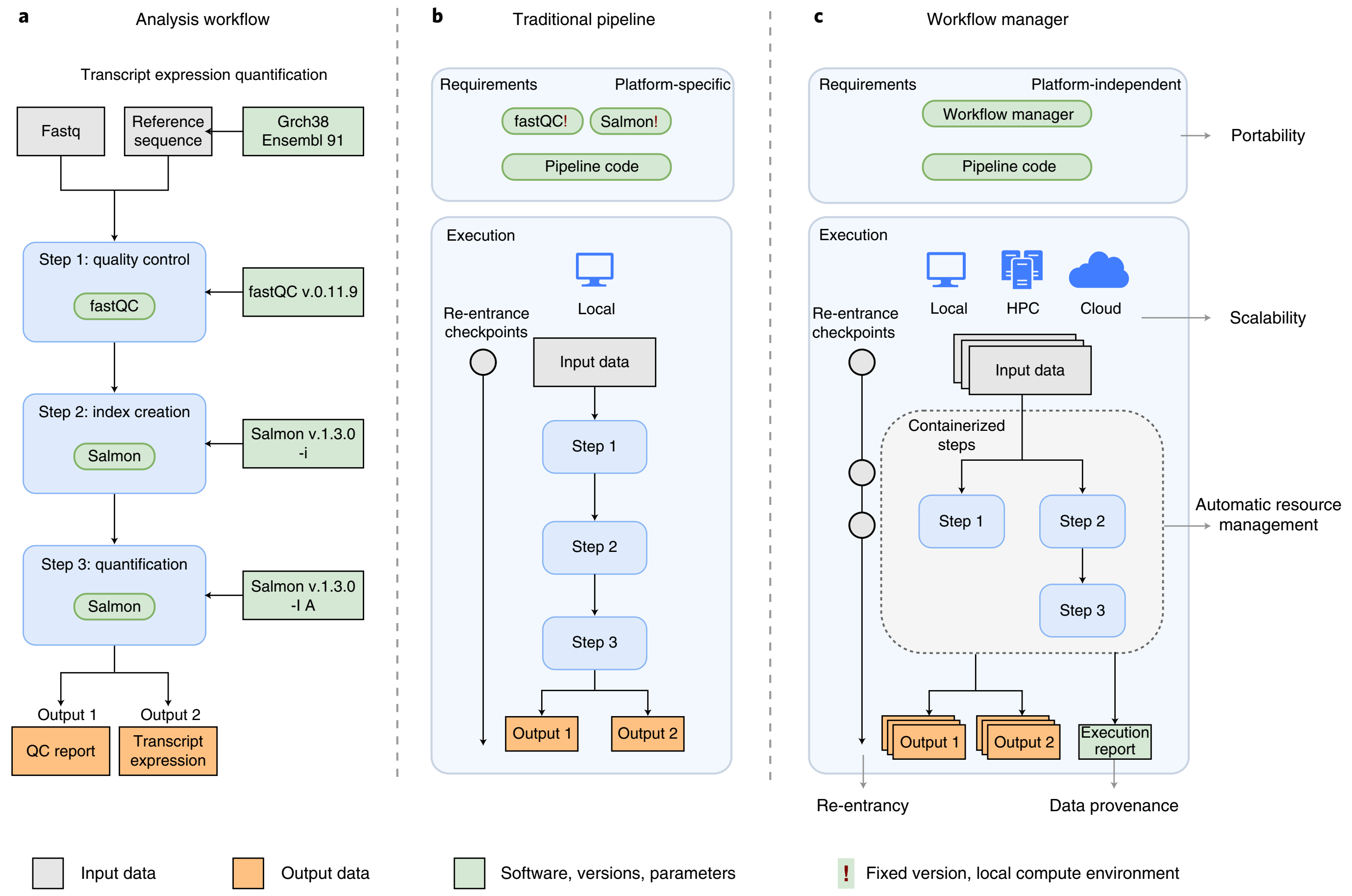

Fig. 1 | Overview of bioinformatics analysis workflows using an example of transcript expression quantification. a, Analysis workflow describing the input data (gray), software, software and reference data versions and parameters (green), and output files (orange). **b**, Traditional pipeline implementations are coupled to the local compute environment and are sensitive to changes in software or data versions. **c**, Implementation of the analysis workflow using a workflow manager decouples the code from the execution environment, enabling portability and more meaningful code sharing. Workflow managers provide options for scalability and optimize resource usage through re-entrance checkpoints and parallelization of steps. Containerized software makes local software installation requirements unnecessary. Execution reports can track parameters and versions, providing transparency and data provenance.

workflow managers[56]. These aspects make Galaxy a powerful choice for users with or without computational expertise who would like to assemble and run custom bioinformatics workflows.

Domain-specific language workflow managers: rapid and flexible development

A domain-specific language (DSL) is a programming language that is developed to meet a specific need within a particular domain[57]. Workflow managers that are implemented as DSLs are developed to enable the rapid deployment of reproducible, robust, and portable computational pipelines and support features specifically for this purpose. Hence, there are many shared features across workflow managers that implement their own DSL, including the ability to incorporate existing tools or pipelines written in other scripting languages, making it easy to port tools over and minimizing refactoring.

Nextflow[6] and Snakemake[58] are popular examples of DSL-based workflow managers that are designed for bioinformaticians familiar with programming. Nextflow, which uses an extension of the Groovy programming language, breaks down each step of a pipeline into modular components and connects these through channels that determine pipeline execution (dataflow paradigm)[6]. In contrast, Snakemake's language is similar to that of standard Python syntax and works backwards by requesting output files and defining each step required to produce them (similar to Make)[59]. Additional examples of DSL-based workflow managers are GenPipes, a Python-based DSL tool for genomics workflows[60], bPipe, which aims for syntactic simplicity using a Groovy-based DSL[61], and Pachyderm, which is used in the banking and automotive industries and also has applications in biotechnology[62]. All of these workflow managers provide a robust and programmatic interface to create pipelines, support containers to ensure portability, and automatically handle resources to optimize scalability (Table 1). A unique feature of Snakemake and Nextflow is the ability to create reusable modules for steps of a workflow. The use of such modules further reduces the complexity of code, increases the readability and maintainability, and makes it easy to extend, reuse, or replace individual steps of analysis workflows (https://snakemake.readthedocs.io/en/stable/snakefiles/modularization.html;https://www.nextflow.io/blog/2020/dsl2-is-here.html). Snakemake also enables between-workflow caching to avoid recomputation of shared steps between pipelines (https://snakemake.readthedocs.io/en/stable/executing/caching.html).

DSL-based workflow managers are well-suited for researchers or teams with prior programming experience. While they provide a powerful framework for rapid pipeline development and minimize the amount of refactoring for pre-existing tools and scripts, the initial learning curve can be steeper than that for graphical workflow managers. To aid this initial learning curve, popular DSL-based workflow managers provide an abundance of learning resources and direct access to community support. Among the DSL-based workflow managers, Nextflow and Snakemake have the largest active community with a large number of published ready-to-use pipelines (for example, refs. [63–66]), making them a popular choice for computational users and teams that want to have maximum flexibility to design custom pipelines.

Programming-library-based workflow managers

While graphical and DSL workflow managers are currently the most widely used frameworks for bioinformatics pipelines, there are some other types of workflow managers, such as programming-library-based tools. Programming-library-based workflow managers implement their pipeline management systems as a programming library for an existing, popular programming language. SciPipe[67], a library for the Go programming language, and Luigi (https://github.com/spotify/luigi), a Python package developed by Spotify, are examples of programming-library-based workflow managers. They leverage existing tooling, text editor support, and other programming libraries in the language[67]. A programming-library-based workflow manager benefits researchers who are already familiar with a programming language and wish to minimize the learning curve of a DSL tool. Compared with other workflow managers, programming-library-based frameworks currently support fewer features and are less adopted for bioinformatics applications (Table 1).

Workflow specifications: portability across workflow systems

Workflow specifications provide a set of formalized rules for defining computational pipelines. This allows the separation of the pipeline definition from the execution environment, thereby adding another layer of abstraction. Workflow specifications enable the definition of pipelines that can be executed across workflow managers or execution environments that support the standard[21]. One example of a workflow specification is the Common Workflow Language (CWL) (https://www.commonwl.org)[68,69]. CWL defines pipelines using YAML (http://yaml.org/) or JavaScript Object Notation (JSON) (http://www.json.org/) formats—human-readable, data-serialization languages. In contrast, the Workflow Description Language (WDL, pronounced 'widdle') (https://openwdl.org/) defines its own human-readable definition language. Execution engines such as Cromwell[70] (https://github.com/broadinstitute/cromwell) and the CWL reference implementation cwltool (https://github.com/common-workflow-language/cwltool) have been developed specifically to run CWL and WDL pipelines, with some existing workflow managers, such as Toil, incorporating CWL and WDL support[71,72]. Some DSL-based workflow managers such as Snakemake implement CWL export functionality (https://snakemake.readthedocs.io/en/stable/executing/interoperability.html)[58].

Workflow specifications such as CWL and WDL are suited to researchers who want to decouple their pipelines from a specific workflow manager to enable a higher degree of portability. Workflow specifications make it easy to define pipelines: they are easy to read and provide the highest level of portability and the most flexible framework for sharing[73]. However, owing to the additional abstraction from separating workflow definition and execution environment, they might appear less convenient than some of the existing workflow managers.

Ready-to-use pipelines provide easy access to complex workflows

The advantages of using workflow managers have contributed to a growing number of publications that provide individual workflow implementations to ensure reproducibility of computational findings. However, most computational pipelines that are developed remain unpublished, with the vast majority likely being reimplementations of similar workflows by different teams or institutions that perform the same analysis[74]. A notable example of this is the Genome Analysis Toolkit Best Practises pipeline[75], which was developed to standardize genomic analysis and now has over 200 implementations on GitHub (https://github.com/search?q=GATK+pipeline&type=Repositories).

To address this, pipeline collections have been developed for all major bioinformatics workflow managers, enabling the easy sharing of pipelines while often dramatically simplifying their execution (Table 2). One of the most extensive collections is hosted by the Galaxy Community project (https://usegalaxy.org/workflows/list_published; https://usegalaxy.eu/workflows/list_published). The Galaxy Community enables anyone to share and execute pipelines, facilitating transparency and reproducibility of scientific workflows. Similarly, KNIME Hub (https://hub.knime.com/) hosts community pipelines for KNIME workflows. However, community pipelines are not required to be peer-reviewed, documented, or maintained.

In contrast, curated pipeline collections have undergone testing and peer review. Such pipelines are often required to have excellent documentation with usage, examples and a description of results, and they are required to be actively maintained by developers. Projects such as snakePipes[76], GenPipes[60], BioWDL (https://biowdl.github.io/) and WARP (https://broadinstitute.github.io/warp/) provide collections of curated pipelines implemented in Snakemake, GenPipes, and WDL, respectively. While these projects ensure that pipelines are maintained and adhere to best practices, they are often smaller and more focused than community-developed pipeline initiatives.

The nf-core project aims to bridge this gap by hosting community-curated pipelines[77]. nf-core provides a framework to host Nextflow pipelines, and it requires specific best practices and sets standards for pipeline implementations to guarantee their maintenance, documentation, portability, scalability, and reproducibility. Among the curated pipeline initiatives, nf-core currently has the largest number of contributors, pipelines, and individual pipeline publications. Snakemake-Workflows is a GitHub repository featuring best-practice, manually reviewed Snakemake workflows with continuous integration testing (https://github.com/snakemake-workflows), while the IWC initiative aims to provide a collection of community-contributed best-practice workflows for Galaxy (https://github.com/galaxyproject/iwc/).

All curated pipeline initiatives enable citability of pipelines and thereby reproducibility of computational analysis[78]. While snakePipes and GenPipes provide a central reference for the pipeline project, nf-core and bioWDL enable the citation of individual pipelines, often with Zenodo DOIs that identify specific release versions (https://zenodo.org/). Curated pipeline projects eliminate the need to redevelop existing pipelines, and they enable researchers without computational experience to use best practices for bioinformatics workflows without the initial learning curve, bringing the power of workflow managers to a much broader audience.

Conclusion

As the need for reproducibility in computational analysis continues to grow, bioinformatics workflow managers have become a key technology. Workflow managers simplify the implementation of robust and complex analysis while providing additional features that help to optimize resource management and reproducibility. Yet, even for noncomputational users, workflow managers might prove to be transformative. Easy-to-use graphical workflow managers, such as Galaxy, and ready-to-use pipeline repositories, like nf-core, enable the execution of complex analysis without programming experience[53,77].

Reproducibility is a major factor for the development of analysis pipelines and can in theory be achieved with well-written custom pipelines. However, workflow managers go beyond the minimal requirements for reproducibility. By improving data provenance, readability, and portability, they increase transparency, enable long-term sustainability of analysis workflows[58], and aid in achieving FAIR (findable, accessible, interoperable, and reusable) computational analysis[22,23]. In particular, the ability to collaboratively develop and share pipelines has led to one of the largest transformations brought by workflow managers. Computational methods are now frequently accompanied with a workflow, core units and bioinformatics teams can make their pipelines available, and international consortia rely on workflow managers for massive data processing.

Community-developed pipelines have been a key contribution toward the increased sharing of code for analysis workflows[79], with curated code repositories enabling the use of best-practice pipelines while providing a template that can be adopted to capture the diversity of tools that often exists. While the use of curated best-practice pipelines ensures robustness, many different best practices might exist for the analysis of biological data. In order to maintain and demonstrate the high standards of public pipelines, benchmarking studies will be needed[80]. While most existing benchmarking studies focus on individual aspects of workflows, benchmarking studies that cover not just individual methods, but also complete workflows, will be essential to evaluate the increasing number of ready-to-use pipelines[81–83].

Future directions

While features and use-cases are currently distinguished by the class of workflow manager, such distinctions will become less clear as workflow manager projects continue to grow and evolve. DSL workflow managers have already introduced graphical user interfaces to deploy and monitor pipelines in the cloud, workflow specifications provide tools for closer integration with existing programming languages, and powerful features and a large community make graphical workflow managers attractive to experienced computational pipeline and methods developers. At the same time, workflow repositories such as WorkflowHub.eu or Dockstore[30], which originated from the complex analysis requirements of large cancer consortia[84], host pipelines for graphical, DSL, and workflow-specification languages, further removing barriers. With a growing number of different workflow managers that support standardized approaches, the performance of the workflow managers themselves will become more relevant. Although benchmarking for scientific software is already common, systematic and quantitative evaluation of the robustness, memory, and storage requirements for different workflow managers will provide further guidance and directions for future developments.

Most bioinformatics software, including workflow managers, is developed as academic open-source projects. While this ecosystem is powerful in providing new solutions, reproducibility of computational analysis workflows requires long-term maintenance[85], code documentation[86], and support for underlying bioinformatics software[87,88]. However, maintaining open-source projects over a long period of time is a major effort[4]. The importance of long-term software maintenance is increasingly appreciated: dedicated funding for essential open-source projects has been made accessible[89] (https://chanzuckerberg.com/), and publicly funded solutions for long-term archiving of data and code are available[90]. By addressing some of the major challenges faced with complex biomedical data processing, workflow managers have already become an essential tool for computational and noncomputational biologists. Combined with initiatives that support and fund best practices for software development and maintenance, standardization, and evaluation, they provide a powerful framework and long-lasting impact to increase the quality and sustainability of code for bioinformatics analysis workflows.

Code availability

Minimal example workflows and links to documentation are available under https://github.com/GoekeLab/bioinformatics-workflows.

Received: 20 October 2020; Accepted: 29 July 2021;
Published online: 23 September 2021

References

1. Stephens, Z. D. et al. Big data: astronomical or genomical? *PLoS Biol.* **13**, e1002195 (2015).
2. Goodwin, S., McPherson, J. D. & McCombie, W. R. Coming of age: ten years of next-generation sequencing technologies. *Nat. Rev. Genet.* **17**, 333–351 (2016).
3. Dozmorov, M. G. GitHub statistics as a measure of the impact of open-source bioinformatics software. *Front. Bioeng. Biotechnol.* **6**, 198 (2018).
4. Nowogrodzki, A. How to support open-source software and stay sane. *Nature* **571**, 133–134 (2019).
5. Mangul, S. et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. *PLoS Biol.* **17**, e3000333 (2019).
6. Di Tommaso, P. et al. Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
7. Tiwari, K. et al. Reproducibility in systems biology modelling. *Mol. Syst. Biol.* **17**, e9982 (2021).
8. Botvinik-Nezer, R. et al. Variability in the analysis of a single neuroimaging dataset by many teams. *Nature* **582**, 84–88 (2020).
9. Grüning, B. et al. Practical computational reproducibility in the life sciences. *Cell Syst.* **6**, 631–635 (2018).
10. van Vliet, M. Seven quick tips for analysis scripts in neuroimaging. *PLoS Comput. Biol.* **16**, e1007358 (2020).
11. Leipzig, J. A review of bioinformatic pipeline frameworks. *Brief. Bioinform.* **18**, 530–536 (2017).
12. Gronenschild, E. H. B. M. et al. The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements. *PLoS ONE* **7**, e38234 (2012).
13. Stodden, V., Seiler, J. & Ma, Z. An empirical analysis of journal policy effectiveness for computational reproducibility. *Proc. Natl Acad. Sci. USA* **115**, 2584–2589 (2018).
14. Reiter, T. et al. Streamlining data-intensive biology with workflow systems. *Gigascience* **10**, giaa140 (2021).
15. Perkel, J. M. Workflow systems turn raw data into scientific knowledge. *Nature* **573**, 149–150 (2019).
16. Love, M. I. et al. Tximeta: reference sequence checksums for provenance identification in RNA-seq. *PLoS Comput. Biol.* **16**, e1007664 (2020).
17. Simoneau, J. & Scott, M. S. In silico analysis of RNA-seq requires a more complete description of methodology. *Nat. Rev. Mol. Cell Biol.* **20**, 451–452 (2019).
18. Simoneau, J., Dumontier, S., Gosselin, R. & Scott, M. S. Current RNA-seq methodology reporting limits reproducibility. *Brief. Bioinform.* **22**, 140–145 (2019).
19. Simoneau, J., Gosselin, R. & Scott, M. S. Factorial study of the RNA-seq computational workflow identifies biases as technical gene signatures. *NAR Genom. Bioinform.* **2**, lqaa043 (2020).
20. Kim, Y.-M., Poline, J.-B. & Dumas, G. Experimenting with reproducibility: a case study of robustness in bioinformatics. *Gigascience* **7**, giy077 (2018).
21. Kanwal, S., Khan, F. Z., Lonie, A. & Sinnott, R. O. Investigating reproducibility and tracking provenance—a genomic workflow case study. *BMC Bioinformatics* **18**, 337 (2017).
22. Goble, C. et al. FAIR Computational Workflows. *Data Intell.* **2**, 108–121 (2020).
23. Lamprecht, A.-L. et al. Towards FAIR principles for research software. *Data Sci.* **3**, 37–59 (2019).
24. Abate, P., Di Cosmo, R., Treinen, R. & Zacchiroli, S. A modular package manager architecture. *Inf. Softw. Technol.* **55**, 459–474 (2013).
25. Decan, A., Mens, T. & Grosjean, P. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empir. Softw. Eng.* **24**, 381–416 (2019).
26. Gruening, B. et al. Recommendations for the packaging and containerizing of bioinformatics software. *F1000Res.* **7**, J-742 (2018).
27. Grüning, B. et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods* **15**, 475–476 (2018).
28. Silver, A. Software simplified. *Nature* **546**, 173–174 (2017).
29. Kurtzer, G. M., Sochat, V. & Bauer, M. W. Singularity: scientific containers for mobility of compute. *PLoS ONE* **12**, e0177459 (2017).
30. O'Connor, B. D. et al. The Dockstore: enabling modular, community-focused sharing of Docker-based genomics tools and workflows. *F1000Res.* **6**, 52 (2017).
31. da Veiga Leprevost, F. et al. BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* **33**, 2580–2582 (2017).
32. Beaulieu-Jones, B. K. & Greene, C. S. Reproducibility of computational workflows is automated using continuous analysis. *Nat. Biotechnol.* **35**, 342–346 (2017).

33. Black, A., MacCannell, D. R., Sibley, T. R. & Bedford, T. Ten recommendations for supporting open pathogen genomic analysis in public health. *Nat. Med.* **26**, 832–841 (2020).
34. Krumm, N. & Hoffman, N. Practical estimation of cloud storage costs for clinical genomic data. *Pract. Lab. Med.* **21**, e00168 (2020).
35. Yang, A., Troup, M. & Ho, J. W. K. Scalability and validation of big data bioinformatics software. *Comput. Struct. Biotechnol. J.* **15**, 379–386 (2017).
36. Krissaane, I. et al. Scalability and cost-effectiveness analysis of whole genome-wide association studies on Google Cloud Platform and Amazon Web Services. *J. Am. Med. Inform. Assoc.* **27**, 1425–1430 (2020).
37. Larsonneur, E. et al. Evaluating workflow management systems: a nioinformatics use case. in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* 2773–2775 (IEEE, 2018).
38. Bux, M. & Leser, U. Parallelization in scientific workflow management systems. Preprint at https://arxiv.org/abs/1303.7195 (2013).
39. Belcastro, L., Marozzo, F. & Talia, D. Programming models and systems for big data analysis. *Int. J. Parallel Emergent Distrib. Syst.* **34**, 632–652 (2019).
40. Silva, V. et al. Raw data queries during data-intensive parallel workflow execution. *Future Gener. Comput. Syst.* **75**, 402–422 (2017).
41. Grossman, R. L. Data lakes, clouds, and commons: a review of platforms for analyzing and sharing genomic data. *Trends Genet.* **35**, 223–234 (2019).
42. Langmead, B. & Nellore, A. Cloud computing for genomic data analysis and collaboration. *Nat. Rev. Genet.* **19**, 325 (2018).
43. Lau, J. W. et al. The Cancer Genomics Cloud: collaborative, reproducible, and democratized—a new paradigm in large-scale computational research. *Cancer Res.* **77**, e3–e6 (2017).
44. Yakneen, S. et al. Butler enables rapid cloud-based analysis of thousands of human genomes. *Nat. Biotechnol.* **38**, 288–292 (2020).
45. Perez-Riverol, Y. & Moreno, P. Scalable data analysis in proteomics and metabolomics using BioContainers and workflows engines. *Proteomics* **20**, e1900147 (2020).
46. Fjukstad, B., Dumeaux, V., Hallett, M. & Bongo, L. A. Reproducible data analysis pipelines for precision medicine. in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)* 299–306 (IEEE, 2019).
47. Birger, C. et al. FireCloud, a scalable cloud-based platform for collaborative genome analysis: strategies for reducing and controlling costs. Preprint at *bioRxiv* https://doi.org/10.1101/209494 (2017).
48. Han, L., Canon, L., Casanova, H., Robert, Y. & Vivien, F. Checkpointing workflows for fail-stop errors. *IEEE Trans. Comput.* **67**, 1105–1120 (2018).
49. Jackson, M., Kavoussanakis, K. & Wallace, E. W. J. Using prototyping to choose a bioinformatics workflow management system. *PLoS Comput. Biol.* **17**, e1008622 (2021).
50. Goecks, J., Nekrutenko, A., Taylor, J. & Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* **11**, R86 (2010).
51. Fillbrunn, A. et al. KNIME for reproducible cross-domain analysis of life science data. *J. Biotechnol.* **261**, 149–156 (2017).
52. Berthold, M. R. et al. in *Data Analysis, Machine Learning and Applications* 319–326 (Springer, 2008).
53. Afgan, E. et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res.* **46**, W537–W544 (2018).
54. Batut, B. et al. Community-driven data analysis training for biology. *Cell Syst.* **6**, 752–758 (2018).
55. Jalili, V. et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update. *Nucleic Acids Res.* **48**, W395–W402 (2020).
56. Ramírez, F. et al. deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic Acids Res.* **44**, W160–W165 (2016).
57. Cordasco, G., D'Auria, M., Negro, A., Scarano, V. & Spagnuolo, C. Toward a domain-specific language for scientific workflow-based applications on multicloud system. *Concurr. Comput.* e5802 (2020).
58. Mölder, F. et al. Sustainable data analysis with Snakemake. *F1000Res.* **10**, 33 (2021).
59. Köster, J. & Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* **28**, 2520–2522 (2012).
60. Bourgey, M. et al. GenPipes: an open-source framework for distributed and scalable genomic analyses. *Gigascience* **8**, giz037 (2019).
61. Sadedin, S. P., Pope, B. & Oshlack, A. Bpipe: a tool for running and managing bioinformatics pipelines. *Bioinformatics* **28**, 1525–1526 (2012).
62. Novella, J. A. et al. Container-based bioinformatics with Pachyderm. *Bioinformatics* **35**, 839–846 (2019).
63. Kieser, S., Brown, J., Zdobnov, E. M., Trajkovski, M. & McCue, L. A. ATLAS: a Snakemake workflow for assembly, annotation, and genomic binning of metagenome sequence data. *BMC Bioinformatics* **21**, 257 (2020).
64. Hölzer, M. & Marz, M. PoSeiDon: a Nextflow pipeline for the detection of evolutionary recombination events and positive selection. *Bioinformatics* **37**, 1018–1020 (2020).
65. Zhao, Q. et al. LncPipe: a Nextflow-based pipeline for identification and analysis of long non-coding RNAs from RNA-seq data. *J. Genet. Genomics* **45**, 399–401 (2018).
66. Cornwell, M. et al. VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. *BMC Bioinformatics* **19**, 135 (2018).
67. Lampa, S., Dahlö, M., Alvarsson, J. & Spjuth, O. SciPipe: a workflow library for agile development of complex and dynamic bioinformatics pipelines. *Gigascience* **8**, giz044 (2019).
68. Amstutz, P. et al. Common Workflow Language v1. 0 (2016); https://doi.org/10.6084/m9.figshare.3115156.v2
69. Crusoe, M. R. et al. Methods included: standardizing computational reuse and portability with the common workflow language. Preprint at https://arxiv.org/abs/2105.07028 (2021).
70. Voss, K., Van der Auwera, G. & Gentry, J. Full-stack genomics pipelining with GATK4 + WDL + Cromwell. *F1000Res* **6**, 1381 (2017).
71. Vivian, J. et al. Toil enables reproducible, open source, big biomedical data analyses. *Nat. Biotechnol.* **35**, 314–316 (2017).
72. Kotliar, M., Kartashov, A. V. & Barski, A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. *Gigascience* **8**, giz084 (2019).
73. Yang, J. Cloud computing for storing and analyzing petabytes of genomic data. *J. Ind. Inf. Integr.* **15**, 50–57 (2019).
74. Xu, B., An, L., Thung, F., Khomh, F. & Lo, D. Why reinventing the wheels? An empirical study on library reuse and re-implementation. *Empir. Softw. Eng.* **25**, 755–789 (2020).
75. McKenna, A. et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).
76. Bhardwaj, V. et al. snakePipes: facilitating flexible, scalable and integrative epigenomic analysis. *Bioinformatics* **35**, 4757–4759 (2019).
77. Ewels, P. A. et al. The nf-core framework for community-curated bioinformatics pipelines. *Nat. Biotechnol.* **38**, 276–278 (2020).
78. Sicilia, M.-A., García-Barriocanal, E. & Sánchez-Alonso, S. Community curation in open dataset repositories: insights from Zenodo. *Procedia Comput. Sci.* **106**, 54–60 (2017).
79. Leman, J. K. et al. Better together: elements of successful scientific software development in a distributed collaborative community. *PLoS Comput. Biol.* **16**, e1007507 (2020).
80. Weber, L. M. et al. Essential guidelines for computational method benchmarking. *Genome Biol.* **20**, 125 (2019).
81. Marx, V. Bench pressing with genomics benchmarkers. *Nat. Methods* **17**, 255–258 (2020).
82. Angers-Loustau, A. et al. The challenges of designing a benchmark strategy for bioinformatics pipelines in the identification of antimicrobial resistance determinants using next generation sequencing technologies. *F1000Res.* **7**, J-459 (2018).
83. Möller, S. et al. Robust cross-platform workflows: how technical and scientific communities collaborate to develop, test and share best practices for data analysis. *Data Sci. Eng.* **2**, 232–244 (2017).
84. Carey, V. J. et al. Global alliance for genomics and health meets Bioconductor: toward reproducible and agile cancer genomics at Cloud scale. *JCO Clin. Cancer Inf.* **4**, 472–479 (2020).
85. List, M., Ebert, P. & Albrecht, F. Ten simple rules for developing usable software in computational biology. *PLoS Comput. Biol.* **13**, e1005265 (2017).
86. Karimzadeh, M. & Hoffman, M. M. Top considerations for creating bioinformatics software documentation. *Brief. Bioinform.* **19**, 693–699 (2018).
87. Anzt, H. et al. An environment for sustainable research software in Germany and beyond: current state, open challenges, and call for action. *F1000Res.* **9**, 295 (2020).
88. Mangul, S., Martin, L. S., Eskin, E. & Blekhman, R. Improving the usability and archival stability of bioinformatics software. *Genome Biol.* **20**, 47 (2019).
89. Siepel, A. Challenges in funding and developing genomic software: roots and remedies. *Genome Biol.* **20**, 147 (2019).
90. Malone, K. & Wolski, R. Doing data science on the shoulders of giants: the value of open source software for the data science community. *Harvard Data Science Review* https://hdsr.mitpress.mit.edu/pub/xsrt4zs2/release/4 (31 May 2020).

Acknowledgements

J.G. is supported by funding from the Agency for Science, Technology, and Research (A∗STAR), Singapore, and by the Singapore Ministry of Health's National Medical Research Council under its Individual Research Grant funding scheme. L.W. was supported by the Singapore International Pre-Graduate Award (SIPGA) from A*STAR and the New Colombo Plan Scholarship from the Australian Department of Foreign Affairs and Trade. We thank B. Grüning for helpful comments and suggestions on this manuscript. We would like to thank R. Patro for contributing a test dataset for the example workflow implementations. We thank M. van den Beek for contributing the Galaxy workflow to the GitHub repository. We thank J. Köster for contributing the Snakemake workflow to the GitHub repository. We thank P. Di Tommaso for contributing the Nextflow workflow to the GitHub repository. We thank S. Lampa for contributing the SciPipe workflow to the GitHub repository. We thank J.H. Gálvez López, P.-O. Quirion, E. Henrion, and M. Bourgey for contributing the GenPipes workflow to the GitHub repository. We thank A. Novak, B. Paten, L. Blauvelt, and L. Koziol for contributing the Toil workflow to the GitHub repository. We thank S. Sadedin for contributing the Bpipe workflow to the GitHub repository. We thank S. Sadedin for contributing the Bpipe workflow to the GitHub repository.

Author contributions

L.W., A.W., and J.G. planned the manuscript. L.W. and J.G. wrote the first draft. L.W., A.W., and J.G. wrote and revised the final manuscript.

Competing interests

A.W. is an employee of ImmunoScape Pte Ltd. L.W. and J.G. declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-021-01254-9.

Correspondence should be addressed to Jonathan Göke.

Peer review information *Nature Methods* thanks Johannes Köster, Yasset Perez-Riverol, Anton Nekrutenko, and Paolo Di Tommaso for their contribution to the peer review of this work. Lin Tang was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.