OXFORD

(GIGA)n SCIENCE

GigaScience, 2018, 1–23

doi: xx.xxxx/xxxx
Manuscript in Preparation
Paper

PAPER

CWLProv: Interoperable Retrospective Provenance Capture and Computational Analysis Sharing

Farah Zaib Khan[1,2,*], Stian Soiland-Reyes[2,3,*], Richard O. Sinnott[1,*], Andrew Lonie[1,*], Carole Goble[3,*] and Michael R. Crusoe[2,*]

[1]The University of Melbourne, Australia and [2]Common Workflow Language Project and [3]The University of Manchester

[*]khanf1@unimelb.edu.au; soiland-reyes@manchester.ac.uk; rsinnott@unimelb.edu.au; alonie@unimelb.edu.au; carole.goble@manchester.ac.uk; mrc@commonwl.org

Abstract

Background: The automation of data analysis in the form of *scientific workflows* has become a widely adopted practice in many fields of research. Computationally driven data-intensive experiments using workflows enable Automation, Scaling, Adaption and Provenance support (ASAP). However, there are still several challenges associated with the effective sharing, publication, understandability and reproducibility of such workflows due to the incomplete capture of provenance and lack of interoperability between different technical (software) platforms.
Results: Based on best practice recommendations identified from literature on workflow sharing and publishing, we define four hierarchical levels of provenance that collectively result in comprehensive and fully re-executable workflows when used with domain-specific information. To realise these levels, we present *CWLProv*, a standard-based format to represent any workflow-based computational analysis to produce workflow output artefacts that satisfy the various levels of provenance. We utilise open source community-driven standards; interoperable workflow definitions in Common Workflow Language (CWL), structured provenance representation using the W3C PROV model, and resource aggregation and sharing as workflow-centric Research Objects (RO) generated along with the final outputs of a given workflow enactment. We illustrate this approach through a practical demonstration of *CWLProv* applied to real-life genomic workflows developed by independent groups.
Conclusions: Our approach to workflow sharing and publication mitigates *workflow decay*. The underlying principles of the standards utilised by *CWLProv* enable semantically-rich and executable Research Objects that capture computational workflows with retrospective provenance such that any platform supporting CWL will be able to understand the analysis, re-use the methods for partial re-runs, or reproduce the analysis to validate the published findings.

Key words: Provenance; Common Workflow Language; CWL; Research Object; RO; BagIt; Interoperability; Portability; Scientific Workflows; Containers

Introduction

Out of the many big data domains, genomics is considered *"the most demanding"* with respect to all stages of the data lifecycle - from acquisition, storage, distribution and analysis [1]. As genomic data is growing at an unprecedented rate due to improved sequencing technologies and reduced cost, it is currently challenging to analyse the data at a rate matching its production. With data growing exponentially in size and volume, the practice to perform computational analyses using *workflows* has overtaken more traditional research methods using ad-hoc scripts that has been the typical modus operandi over the last few decades [2, 3]. Scientific workflow design and management has become an essential part of many computationally

Compiled on: October 28, 2018.
Draft manuscript prepared by the author.

Key Points
The contribution of this paper is fourfold

- We have gathered best-practice recommendations that we identified in existing literature, and reflect on the various authors' experiences with workflow managements systems and especially with regards to factors to consider when a computational analysis is designed, executed and shared.
- Combining the above with our own experiences from empirical studies [6, 7, 8, 9], we define a set of hierarchical levels of provenance tracking and method sharing where the highest level represent complete understanding of the shared resources supported by reproducibility and re-use of the methods from the lower levels.
- Building on this provenance hierarchy, we define *CWLProv* for the methodical representation of artefacts associated with a given workflow enactment associated with any study involving computational data-intensive analysis.
- Finally, we demonstrate the utilisation of *CWLProv* by extending an existing workflow execution engine *cwltool* [10] to produce workflow-centric Research Objects generated as a result of a given workflow enactment. We illustrate this through a case study of using workflows designed by external (independent) developers, and subsequently evaluate the interoperability, portability and completeness of the generated CWLProv outcome.

driven data-intensive analyses enabling Automation, Scaling, Adaptation and Provenance support (ASAP)[4]. Increased use of workflows has driven rapid growth in the number of computational data analysis workflow systems, with hundreds of heterogeneous approaches now existing for workflow specification and execution [5]. The need for a common format and standard to define workflows and enable sharing of analysis results using a given workflow environment is urgently required.

Common Workflow Language (CWL) [11] has emerged as a workflow definition standard designed to enable portability, interoperability and reproducibility of analyses between workflow platforms. CWL has been widely adopted by more than 20 organisations, providing an interoperable bridge overcoming the heterogeneity of workflow environments. Whilst a common standard for workflow definition is an important step towards interoperable solutions for workflow specifications, sharing and publishing the results of these workflow enactments in a common standardised format is also equally important. The transparent and comprehensive sharing of experimental designs is critical to establish trust and ensure authenticity, quality and reproducibility of any workflow-based research result. Currently there is no common format defined and agreed upon for interoperable workflow archiving or sharing [12].

In this paper, we utilise open-source standards such as CWL together with related efforts such as Research Objects (ROs) [13], BagIt [14] and PROV [15] to define ***CWLProv***, a format for the interoperable representation of a CWL workflow enactment. We focus on production of a workflow-centric executable RO as the final result of a given CWL workflow enactment. This RO is equipped with all of the artefacts used in a given execution including the workflow inputs, outputs and, most importantly, the retrospective provenance. This approach enables the complete sharing of a computational analysis such that any future CWL-based workflow can be re-run given the best practices discussed later for software environment provision are followed.

The concept of workflow-centric ROs has been previously considered [13, 16, 17] for structuring the analysis methods and aggregating digital resources utilized in a given analysis. The generated ROs in these studies typically aggregate data objects, example inputs, workflow specifications, attribution details, details about the execution environment amongst various other elements. These previous efforts were largely tied to a single platform or a single workflow management system. *CWLProv* aims to provide a platform-independent solution for workflow sharing, enactment and publication. The standards and vocabularies used to design *CWLProv* all have an overarching goal to support a domain-neutral and interoperable solution (detailed in Section **Applied Standards and Vocabularies**).

The contribution of this work are summarised and listed in the **Key Points** section and the remainder of this paper is structured as follows. In Section **Background and Related Work** we discuss the key concepts and related work followed by a summary of the published best-practices and recommendations for workflow representation and sharing in Section **Levels of Provenance and Resource Sharing**. This section also details the levels of provenance capture and method sharing we define. Section **CWLProv and utilised standards** introduces *CWLProv* and outlines its format, structure and the details of the standards and ontologies it utilises. Section **Practical Realisation of CWLProv** presents the implementation details of *CWLProv* using *cwltool* [10] and Section **CWLProv Evaluation with Bioinformatics Workflows** demonstrates and evaluates the implemented module for three existing workflow case studies. We discuss the challenges of interoperable workflow sharing and the limitations of the proposed solution listing several possible future research directions in Section **??** before finally drawing conclusions on the work as a whole in Section **Conclusion**.

Background and Related Work

This work draws upon a range of topics: *Provenance*, *Interoperability* and *Portability*. We define these here to provide better context for the reader.

Provenance

A number of studies have advocated the need for complete provenance tracking of scientific workflows to ensure transparency, reproducibility, analytic validity, quality assurance and attribution of (published) research results [18]. The term *Provenance* is defined by World Wide Web Consortium (W3C) [19] as:

> *"Provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness."*

Provenance for workflows is commonly divided into the following three categories: *Retrospective Provenance*; *Prospective Provenance* and *Workflow Evolution*. *Retrospective Provenance* refers to the detailed record of the implementation of a computational task including the details of every executed process together with comprehensive information about the execution environment used to derive a specific product. *Prospective Provenance*

refers to the 'recipes' used to capture a set of computational tasks and their order, e.g. the workflow specification [20]. This is typically given as an abstract representation of the steps (tools/data analysis steps) that are necessary to create a particular research output, e.g. a data artefact. *Workflow Evolution* refers to tracking of any alteration in the existing workflow resulting in another version of the workflow that may produce either the same or different resultant data artefacts [21]. In this work, our focus is mainly on *Retrospective Provenance*.

Interoperability

The concept of interoperability varies in different domains. Here we focus on *computational interoperability*. This is defined as:

> *The ability of two or more components or systems to exchange information and to use the information that has been exchanged* [22].

The focus of this study is to achieve *syntactic*, *semantic* and *pragmatic* interoperability as defined in Levels of Conceptual Interoperability Model (LCIM)[23]. *Syntactic* interoperability is achieved when a common data format for information exchange is unambiguously defined. The next level of interoperability, referred to as *semantic* interoperability, is reached when the content of the actual information exchanged is unambiguously defined. Once there is an agreement about the format and content of the information, *pragmatic* interoperability is achieved when the context, application and use of the shared information and data exchanged is also unambiguously defined. In Section **Evaluation Results**, we relate these general definitions to specific workflow applications with respect to workflow-centric ROs.

Portability

In the context of computing, *Portability* is defined as:

> *The ability of software to be transferred from one machine or system to another* [24].

The concept of portability of software can relate to the underlying hardware resources as well as the software environment and any dependencies and availability of resources required for the software to function, e.g. the operating system. In this work, we consider portability with respect to the software environment.

Related Work

We focus on relevant studies and efforts trying to resolve the issue of availability of required resources used in a published article or a given computational analysis. In addition, we cover efforts directed towards provenance capture of workflow enactments. As these concepts have been around for a considerable time, we restrict our attention to scientific workflows and studies related to the biological domain.

Freezing and packaging the run-time environment to encompass all the software components and their dependencies used in an analysis is a recommended and widely adopted practice [25] especially after frequent use of cloud computing resources where images and snapshots of the cloud instances are created and shared with fellow researchers [?]. Nowadays, preservation and sharing of the software environment e.g. in open access repositories, is becoming a regular practice in the workflow domain as well. Leading platforms managing infrastructure and providing cloud computing services and configuration on demand include DigitalOcean, Amazon Elastic Compute Cloud (EC2), Google Cloud Platform, Microsoft Azure and Rackspace Cloud Servers. The instances launched on these platforms can be saved as snapshot and published with an analysis study to later create an instance representing the computing state of the snapshot.

Using *"System-wide packaging"* for data-driven analyses, although simplest on part of the workflow developers and researchers, has its own caveats. One of the notable issue is the size of the snapshot as it captures everything in an instance at a given time, hence the size can range from few gigabytes to many terabytes. To address this issue, various lightweight container-based virtualisation solutions and software package managers are emerging as a solution to distribute research software and share execution environments including efforts such as: Docker[26], Singularity[27], Debian Med[28], and Bioconda[29]. Docker is a lightweight container-based virtualisation technology that facilitates the automation of application development by archiving software systems and environment. Docker images (an immutable source of containers) can be composed of multiple layers and shared publicly, e.g. through the open-source Docker hub [30] thereby making the software environment required for any analysis available. Docker improves portability of applications as it can run on many common platforms including Linux, Microsoft windows, Mac OS and Cloud instances. Singularity is also a cross-platform open source container-based initiative specifically supporting High Performance Computing (HPC) resources. An existing Docker image can be imported and used by Singularity. Bioconda, based on the an open source package manager Conda [31], is directed towards availability and portability of software used in the life science domain. Bioconda packages are portable across Microsoft Windows, Mac OS and Linux environments. The Debian Med community creates high quality packages of bioinformatic software for the Debian Linux distribution with an emphasis on co-installability and cross-testing. In addition, each package can be installed using multiple package managers including Conda and Debian's apt with Docker and Singularity offering a higher degree of flexibility to users.

Preserving and sharing only the software environment is not enough to verify results of a published computational analysis or re-use the methods used (e.g. workflows) with a different dataset. It is also necessary to share other details including data (exemplar or the original), scripts, workflow files, input configuration settings, the hypothesis of the experiment and any/all trace/logging information related to "what happened", i.e. the retrospective provenance of the actual workflow enactment. The publishing of resources to improve state of scholarly publications is now supported by various online repositories. Examples of these include Zenodo [32], GitHub [33], myExperiment [34] and figshare [35]. These resources facilitate collaborative research in addition to public sharing of source code and the results of a given analysis. However, there is no standard format defined or declared that must be followed when someone shares artefacts associated with an analysis. As a result, the quality of the shared resources can range from a highly annotated, properly documented and complete set of artefacts, to raw data with undocumented code and incomplete information about the analysis as a whole. Individual organisations or groups might provide a set of "recommended practices", e.g. in readme files to attempt to maintain the quality of shared resources. An exemplar initiative *Code as a Research Object* [36] is a joint project between Figshare, GitHub and Mozilla Science Lab aimed to archive any GitHub code repository to Figshare and produce a Digital Object Identifier (DOI) to improve the discovery of resources.

Reprozip[37] aims to resolve portability issues by identifying and packaging all dependencies in a self-contained package

which when unpacked and executed on another system (with Reprozip installed) should reproduce the methods and results of the analysis. Each package also contains a human readable configuration file containing provenance information obtained by tracing system calls during system execution. The provenance trace is not formatted using existing open standards established by the community however.

Several platform-dependent studies have been targeted towards extensions to existing standards by implementing Research Object model and improving aggregation of resources. Belhajjame et al. [13] proposed the application of ROs to develop workflow-centric ROs containing data and metadata to support the understandability of the utilised methods (in this case workflow specifications). They explored five essential requirements to workflow preservation and identified data and metadata that could be stored to satisfy the said requirements. They proposed extensions to existing ontologies such as Object Reuse and Exchange (ORE), the Annotation Ontology (AO) and PROV-O ontology and developed four new ontologies to represent workflow specific information. However, as stated in the paper, the scope of the proposed model at that time was not focused on interoperability of heterogeneous workflows as it was demonstrated for a workflow specific to Taverna Workflow Management System using myExperiment, which makes it quite platform-dependent.

A domain-specific solution is proposed by Gomez-Perez et al. [38] by extending the RO model to equip workflow-centric ROs with information catering for the specific needs of the Earth Science community, resulting in enhanced discovery and reusability by experts. They demonstrated that the principles of ROs can support extensions to generate aggregated resources leveraging domain specific knowledge. Hettne et al. [16] used three genomic workflow case studies to demonstrate the utilisation of ROs to capture methods and data supporting querying and useful extraction of information about the scientific investigation under observation. The solution was tightly coupled with the Taverna Workflow Management System and hence if shared, would not be interoperable or reproducible outside of the Taverna environment. Other notable efforts to use ROs for workflow preservation and method aggregation include [7] in systems biology, [39] in clinical settings and [9] in precision medicine.

There are a range of provenance standards that have also been proposed. Many studies have emphasised the need for provenance focusing on aspects such as scalability, granularity, security, authenticity, modelling and annotation [18]. They identify the need to support standardised dialogues to make provenance interoperable. Many of these were used as inputs to initial attempts at creating a standard Provenance Model to tackle the often inconsistent and disjointed terminology related to provenance concepts. This ultimately resulted in the specification of the Open Provenance Model (OPM) v1.00 [40] together with an open-source model for the governance of OPM [41]. Working towards similar goals of interoperability and standardization of provenance for web technologies, the World Wide Web Consortium (W3C) Provenance Incubator Group [42] and the authors of OPM together set the fourth provenance challenge at the International Provenance and Annotation Workshop, 2010 (IPAW'10) that resulted in *PROV*, a family of documents serving as the conceptual model for provenance capture, its representation, and sharing and exchange over the Web [43] regardless of the domain or platform. Since then, a number of studies have proposed extensions to this domain-neutral standard to cater for a range of special circumstances. The model is general enough to be adapted to any field and flexible enough to allow extensions for specialised cases.

Michaelides et al. [44] presented a domain-specific PROV-based solution for retrospective provenance to support portability and reproducibility of a statistical software suite. They captured the essential elements from the log of a workflow enactment and represented them using an intermediate notation. This representation was later translated to PROV-N and used as the basis for the PROV Template System. A Linux specific system provenance approach was proposed in [45] where they demonstrated retrospective provenance capture at the system level. Another ongoing project, UniProv is working to extract information from Unicore middleware and transform it into a PROV-O representation to facilitate the back-tracking of workflow enactments [46]. The toolkit, *PROV-man* also uses the PROV standard to record provenance information. Provenance information can be created, managed and queried using programming APIs and stored in configurable relational databases [47]. Platforms such as VisTrials [48] and Taverna [7] have built in retrospective provenance support. Taverna implements an extensive provenance capture system "TavernaProv", utilising both PROV ontologies as well as ROs aggregating the resources used in an analysis [49]. Vistrails is an open source project supporting platform-dependent provenance capture, visualisation and querying for extraction of required information about a workflow enactment. Chirigati et al. [37] provide an overview of PROV terms and how they can be translated from the VisTrials schema and serialised to PROV-XML.

All these efforts are fairly recent and use a standardised approach to provenance capture and hence are relevant to our work on the capture of retrospective provenance. However, our aim is a domain-neutral and platform-independent solution that can be easily adapted for any domain and shared across different platforms and operating systems.

As evident from the literature, there are efforts in progress to resolve the issues associated with effective and complete sharing of computational analysis including both the results and provenance information. These studies range from highly domain-specific solutions and platform-dependent objects to open source flexible interoperable standards. CWL has widespread adoption as a workflow definition standard, hence is an ideal candidate for portable workflow definitions. The next section investigates existing studies focused on workflow-centric science, and summarises best practice recommendations put forward in these studies. From this we define a hierarchical view of provenance and resource sharing.

Levels of Provenance and Resource Sharing

Various studies have empirically investigated the role of automated computational methods in the form of workflows and published best practice recommendations to support workflow preservation, validity, understandability and re-use. We summarise a number of these recommendations and the justifications (why is it required) in Table 1 with focus on those that have guided our choices of standards to be used in *CWLProv*. These recommendations aid in our understanding to define hierarchical levels of provenance and resource sharing to classify the granularity of the information associated with a workflow enactment in Figure 1. The four levels, Level 0 to Level 3 are described in the following sections and linked to the requirements stated in Table 1 that these levels aim to satisfy.

Level 0

To achieve this level, researchers should share the workflow specifications, input parameters used for a given workflow enactment, raw logs and output data preferably through an open-access repository. This is the least information that could be

Table 1. Summarised recommendations and justifications from various studies covering best practices on reproducibility, accessibility, interoperability and portability of workflows

R.no	Recommendations	Justifications
R1	Save and share all parameters used for each software used in a given workflow (including default values of parameters used) [50, 51, 52, 53].	Impacts on reproducibility of results since different inputs and configurations of the software can produce different results. Different versions of a tool might upgrade the default values of the parameters.
R2	Avoid manual processing of data and if using *shims* [54] then make these part of the workflow to fully automate the computational process [50, 53].	This ensures the complete capture of the computational process without broken links so that the analysis can be executed without need for performing manual steps.
R3	Include intermediate results where possible when publishing an analysis [51, 52, 53].	Intermediate data products can be used to inspect and understand shared analysis when re-enactment is not possible.
R4	Record the exact software versions used [50, 53].	This is necessary for reproducibility of results as different software versions can produce different results.
R5	If using public data (reference data, variant databases), then it is necessary to store and share the actual data versions used [3, 6, 50, 53] .	This is needed as different versions of data, e.g. human reference genome or variant databases, can result in slightly different results for the same workflow.
R6	Annotation tools such as user contributed tags and versions should be assigned to workflows and shared when publishing the workflows and their associated results [55].	Annotations make workflows accessible and enhance their re-use and hence promote the longevity of the workflows.
R7	Workflows should be well-described, annotated and offer associated metadata [13, 17, 52, 55, 56] .	Metadata and annotations improve the understandability of the workflow and facilitate independent re-use by someone skilled in the field.
R8	Use and store stable identifiers for all artefacts including the workflow, the datasets or the software components [55, 56].	Identifiers play an important role in the discovery and accessibility of resources made available in open-access repositories.
R9	Share the details of the computational environment [13, 6, 56] .	Such details support requirements analysis before any re-enactment or reproducibility is attempted.
R10	Share workflow specifications/descriptions used in the analysis [13, 51, 52, 56, 57].	The same workflow specifications can be used with different datasets thereby supporting re-usability.
R11	Aggregate the software with the analysis and share this when publishing a given analysis [13, 6, 56, 57, 52].	Making software available reduces dependence on third party resources and as a result minimizes *workflow decay* [58].
R12	Share raw data used in the analysis [13, 51, 52, 56, 57].	When someone wants to validate published results, availability of data supports verification of claims and hence establishes trust in the published analysis
R13	Store all attributions related to data resources and software systems used [52, 57].	Accreditation supports proper citation of resources used.
R14	Workflows should be preserved along with the provenance trace of the data and results [13, 17, 52, 53, 57].	A provenance trace provides a historical view of the workflow enactment enabling end users to better understand the analysis retrospectively
R15	Data flow diagrams of the computational analysis using workflows should be provided [6, 51].	These diagrams are easy to understand and provide a human readable view of the workflow.
R16	Open source licensing for methods, software, code, workflows and data should be adopted instead of proprietary resources [6, 51, 53, 56, 57, 59].	This will ensure availability of all of the resources used in the original analysis since use of restricted licenses hinder reproducibility.
R17	Data, code and all workflow steps should be shared in a format that others can easily understand preferably in a system neutral language [13, 51, 59].	System neutral languages help achieve interoperability and make an analysis understandable.
R18	Promote easy execution of workflows without making many changes to the underlying environment [3].	This supports the adaption of available analysis methods to new settings and thereby improves portability of published studies.
R19	Information about compute and storage resources should be stored and shared as part of the workflow [6].	Such information can assist users in estimating the required resources needed for an analysis and thereby reduce the amount of failed executions.
R20	Example input and sample output data should be preserved and published along with the workflow-based analysis [13, 58].	This information enables test runs of an analysis to verify the methods used.

This list is not exhaustive and other studies have identified separate issues - provenance and data security, for example - but these are beyond the scope of the work here.

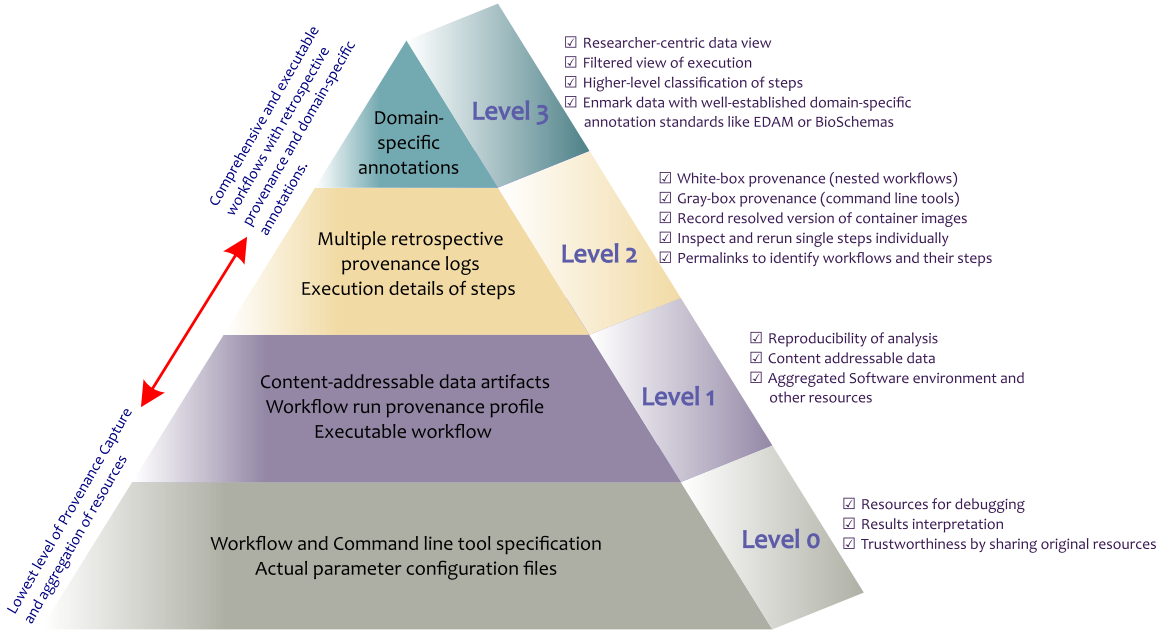

Figure 1. Levels of Provenance and resource sharing and their applications

shared without putting any extra efforts to support seamless reproducibility or understandability of a given analysis. The artefacts shared at this level would only require uploading of the associated resources to a repository without necessarily providing any supporting metadata or provenance information. Information captured at Level 0 is the bare minimum that can be used for result interpretation.

Workflow definitions based on Level 0 can also potentially be re-purposed for other analyses. According to Ludäscher, a well-written scientific workflow and its graphical representation is itself a source of prospective provenance giving user an idea of the steps taken and data produced [60]. Therefore a well-described workflow specification indirectly provides prospective provenance without aiming for it. In addition to the textual workflow specification, its graphical representation should also be shared if available for better understandability fulfilling *R15*. At this level, re-running the workflow would only be possible if the end-user devotes extra efforts to make sense of the shared artefact. As open access journals require availability of methods and data, many published studies now share workflow specifications and optionally the outputs thereby achieving "Level 0" and specifically satisfying *R1* and *R10* (Table 1).

Level 1

At Level 1, *R4*, *R5* and *R14* should be satisfied by providing retrospective provenance of the workflow enactment - i.e. a structured representation of machine readable provenance which can answer questions such as "what happened", "when happened", "what was executed", "what was used", "who did this" and "what was produced". The seamless re-enactment of the workflow should be supported at this level. This is only possible when along with provenance information, *R9* and *R11* is satisfied by potentially packaging the software environment for analysis sharing or there is enough information about the software environment that can guide the user to re-enact the workflow.

In addition to the software availability and retrospective provenance, access to input data should also be provided fulfilling *R12*. This data can either be used to re-enact the published methods or utilised in a different analysis, e.g. for performance comparison of methods. At Level 1, it is preferable to provide content-addressable data artefacts such as input, output and intermediate files, avoiding local paths and file names to make a given workflow executable. The intermediate data artefacts should also be provided where necessary to facilitate inspection of results hence satisfying *R3*.

While software and data can be digitally captured, the hardware and infrastructure requirements also need to be captured to fulfill *R19* and provide a priori estimation of the computational cost and capability needed for re-running or reusing a workflow. This kind of information can naturally vary widely with runtime environments, architectures and data sizes, as well as rapidly becoming outdated as hardware and cloud offerings evolve. Nevertheless a snapshot of the workflow's overall execution resource usage for an actual run can be beneficial to give a broad overview of the requirements.

Level 2

It is a common practice in scientific workflows to modularize the workflow specifications by separating the related tasks into "sub-workflows" or "nested workflows" [25] to be incorporated and used in other workflows or be assigned to appropriate compute and storage resources in case of distributed computing [61]. These modular solutions promote understanding and re-usability of the workflows as researchers are inclined to use these modules instead of workflow as whole for their own computational experiments. An example of a sub-workflow is the mandatory "pre-processing" [62] needed for the Genome Analysis ToolKit (GATK) best practice pipelines used for genomic variant calling. These steps can be separated into a workflow to be used before any variant calling pipeline, be it somatic or germline.

At Level 1, retrospective provenance is coarse grained and as such, there is no distinction between workflows and their sub-workflows. This results in *black-box* provenance as stated by Ludäscher [60] who distinguishes workflow provenance as *black-box* and database provenance as *white-box*. The reasoning behind this distinction is that often the steps in a workflow, especially those based on graphical user interface-based

platforms, provide levels of abstraction/obscurity to the actual tasks being implemented. In our previous work we used an empirical case study to demonstrate that declarative approaches to workflow definition resulted in transparent workflows with the least number of assumptions [6]. This resolves the black box/white box issue to some extent but to further support research transparency, we propose to share retrospective provenance logs for each nested/sub-workflow making the details of a workflow enactment as explicit as possible and moving a step closer to *white-box* provenance. These provenance logs will support the inspection and re-enactment of targeted components of a workflow such as a single step or a sub-workflow individually without necessarily having to re-enact the full analysis.

In addition, we propose to include *permalinks* at Level 2 to identify the workflows and their individual steps which facilitates the inspection of each step and aim to improve the longevity of the shared resources, hence supporting *R8*.

Improving *R19* would for level 2 include resource usage per task execution. Along with execution times this can be useful information to identify bottlenecks in a workflow and for more complex calculation for cost estimation. At this level resource usage data will however also become more noisy and highly variant on scheduling decisions by the workflow engine, e.g. sensitivity to cloud instance reuse or co-use for multiple tasks, or variation in data transfers between tasks on different instances. Thus level 2 resource usage information should be further processed with statistical models for it to be meaningful for a user.

Level 3

Levels 0-2 are generic and domain-neutral, and can apply to any scientific workflow. However, domain-specific information/metadata about data and processes plays an important role in better understanding of the analysis and exploitation of provenance information, e.g. for meaningful queries to extract information to the domain under consideration [63]. Addition of domain specific metadata e.g. file formats, user-defined tags and other annotations to generic retrospective provenance can improve the *white-boxness* by providing domain context to the analysis as described in *R6* and *R7*. Annotations can range from adding textual description and tags to marking data with more systematic and well-defined domain-specific ontologies such as EDAM [64] and BioSchemas [65] in the case of bioinformatic workflows. Some studies also propose to provide example or test data sets which eventually helps in analyzing the methods shared and verifying their results (as described in *R20*).

At Level 3, the information from previous levels combined with specific metadata about data artefacts facilitates higher level classification of workflow steps into motifs [66] such as data retrieval, pre-processing, analysis and visualisation. This level of provenance, resource aggregation and sharing can provide a researcher-centric view of data and enable users to re-enact a set of steps or full workflow by providing filtered and annotated view of the execution. This can be non-trivial to achieve with mainstream methods of workflow definition and sharing, as it requires guided user annotations with controlled vocabularies, but this can be simplified by reusing related tooling from existing efforts like BioCompute Objects [67] and DataCrate [68].

Communicating resource requirements at Level 3 (*R19*) would involve domain-specific models, such as a cost estimation based on number of base pairs or measures of protein complexity. Such models might need to be derived from resource usage across multiple workflow runs with varied inputs, e.g. by a multi-user workflow platform like Seven Bridges Platform. Applying Level 3 resource usage models might require pre-processing workflow inputs and calculations in an environment like R or Python, and so we recommend that models are provided as separate sidecar workflows for interoperable execution before the main workflow.

Requiring researchers to achieve the above defined levels individually is unrealistic without an associated guidance framework. Ideally, the conceptual meaning of these levels would be translated into a practical solution utilising the available resources. In the next section, we propose a format for annotating resource aggregations equipped with retrospective provenance and describe the associated standards that can be applied in this process.

CWLProv and utilised standards

Here we present *CWLProv*, a format for the methodical representation of workflow enactment, associated artefacts and capturing and using retrospective provenance information. Keeping in view the recommendations from Table 1 for example *R16* and *R17*, we leverage **open-source**, **domain-independent**, **system-neutral**, **interoperable** and most importantly **community-driven** standards as the basis for the design and formatting of reproducible and interoperable workflow-based ROs.

Applied Standards and Vocabularies

We follow the recommendation *"Reuse vocabularies, preferably standardised ones"* [69] from best practices associated with data sharing, representation and publication on the web to achieve consensus and interoperability of workflow-based analyses. Specifically we integrate the Common Workflow Language (CWL) for workflow definition, Research Objects (ROs) for resource aggregation and the PROV-Data Model (PROV-DM) to support the retrospective provenance associated with workflow enactment. The key properties and principles of these standards are described below.

- ***Common Workflow Language (CWL)***

CWL provides declarative constructs for workflow and command line tool definition. It makes minimal assumptions about base software dependencies, configuration settings, software versions, parameter settings or indeed the execution environment more generally [6]. CWL supports the comprehensive recording and capture of information that is created during workflow design and execution. This can subsequently be structured and published alongside any resultant analysis using that workflow.

CWL is a community-driven effort that has been widely adopted by many workflow design and execution platforms supporting interoperability across diverse platforms. Examples of current adopters include workflow-centric research efforts such as Toil, Arvados, Rabix [70], Cromwell [71], REANA, and Bcbio [72] with implementations for Galaxy, Apache Taverna, and AWE currently in progress. CWL's object model supports the declaration of information which can be captured, structured and saved as provenance when a workflow is enacted.

A workflow in CWL is composed of "steps" where each step refers either to a command line tool (also specified using CWL) or another workflow specification incorporating the concept of "sub-workflows". Each "step" is associated with "inputs" that are comprised of any data artefact required for the execution of that step (Figure 2). As a result of the execution of each step, "outputs" are produced which can become (part of) "inputs" for the next steps making the execution data-flow oriented.

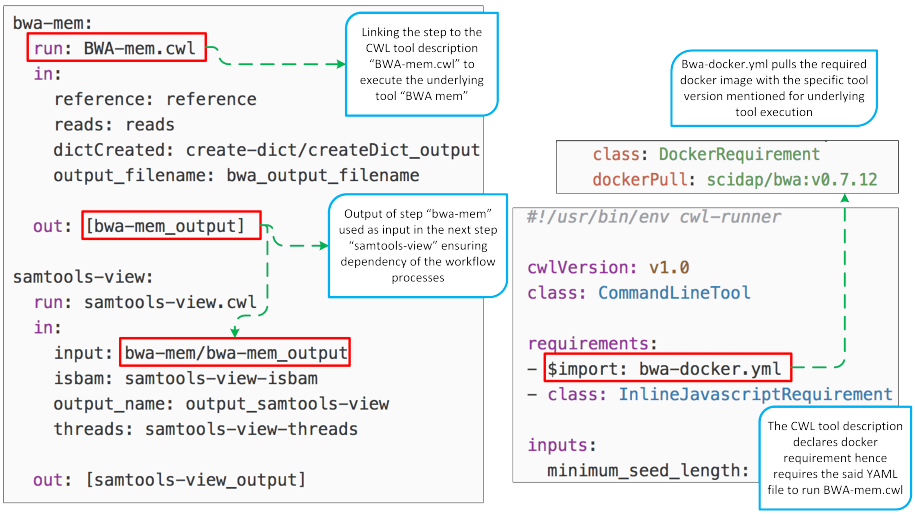

Figure 2. Right: A snapshot of part of a GATK workflow described using CWL. Two steps named as *bwa-mem* and *samtools-view* are shown where the former links to the tool description executing the underlying tool (BWA-mem for alignment) and provides the output used as input for samtools. Left: Snapshot of BWA-mem.cwl and the associated Docker requirements for the exact tool version used in the workflow execution.

CWL is not tied to a specific operating system or platform which makes it an ideal approach for interoperable workflow definitions.

• ***Research Object (RO)***

An RO encapsulates all of the digital artefacts associated with a given computational analysis contributing towards preservation of the analysis.

The aggregated resources can include but are not limited to: input and output data for analysis results validation; computational methods such as command line tools and workflow specifications to facilitate workflow re-enactment; attribution details regarding users; retrospective as well as prospective provenance for better understanding of workflow requirements, and machine-readable annotations related to the artefacts and the relationships between them. The goal of ROs is to make any published scientific investigation and the produced artefacts *"interoperable, reusable, citable, shareable and portable"*.

The three core principles [73] of the RO approach are to support "Identity", "Aggregation", and "Annotation" of research artefacts. They look to enable accessibility of tightly-coupled, interrelated and well-understood aggregated resources involved in a computational analysis as identifiable objects, e.g. using unique (persistent) identifiers such as DOIs and/or ORCIDs. The RO approach is well aligned with the idea of interoperable and platform-independent solutions for provenance capture of workflows because of its domain-neutral and platform-independent nature.

While ROs can be serialised in several different ways, in this work we have reused the BDBag approach based on *BagIt*, which has been shown to support large-scale workflow data [74]. This approach is also compatible with data archiving efforts from the NIH Data Commons, Library of Congress and the Research Data Alliance. The specialised workflow-centric RO in this study encompasses the components mentioned in the previous paragraph annotated with various targeted tools and a PROV-based *"Workflow provenance profile"* to capture the detailed retrospective provenance of the CWL workflow enactment.

• ***PROV Data Model (PROV-DM)***

The World Wide Web Consortium (W3C) developed PROV based on the recommendations of the Provenance Incubator Group. PROV refers to a suite of specifications introduced to support the unified/interoperable representation and publication of provenance information on the Web. The underlying conceptual PROV Data Model (PROV-DM), provides a domain-agnostic model designed to capture fundamental features of provenance with support for extensions to integrate domain-specific information (Figure 3).

We utilise mainly two serialisations of PROV for this study, PROV-Notation (PROV-N) [76] and PROV-JSON [77]. PROV-N is designed to achieve serialisation of PROV-DM instances by formally representing the information using simplified technology-independent syntax to improve readability. PROV-JSON is a lightweight interoperable representation of PROV assertions using JavaScript constructs and data types and can be used further for querying the information. The key design and implementation principles of these two serialisations of PROV are in compliance with the goals of this study i.e. understandable and interoperable, hence are a natural choice to support the design of an adaptable provenance profile.

Framework of CWLProv Research Object

The levels of provenance and resource sharing defined in previous section can be satisfied by using a structured approach to share the identified resources. In this section, we define

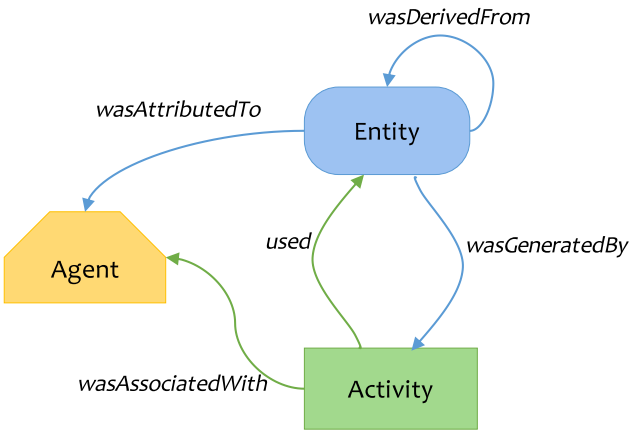

Figure 3. Core concepts of the PROV Data Model. Adapted from W3C PROV Model Primer [75].

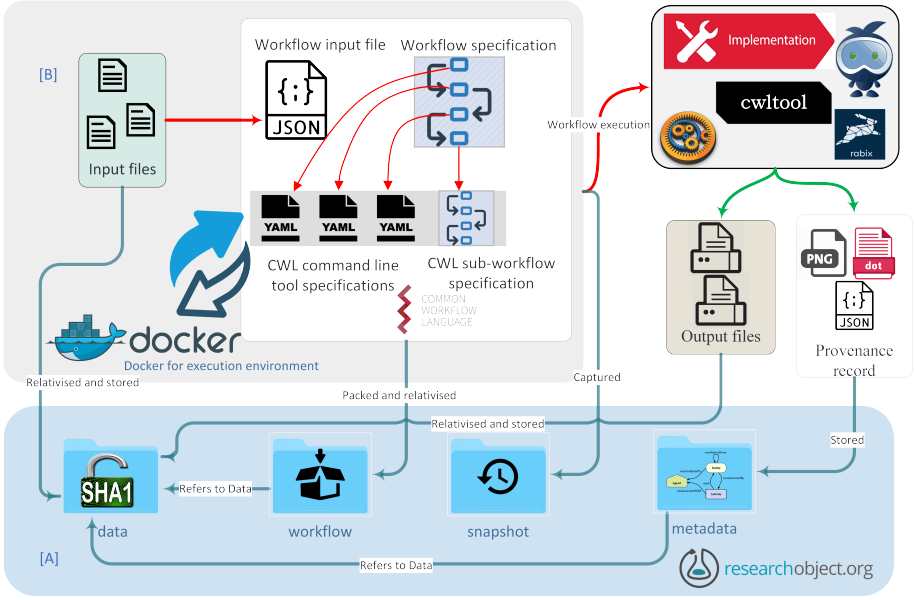

Figure 4. Schematic representation of the aggregation and links between the components of a given workflow enactment. Layers of execution are separated for clarity. The workflow specification and command line tool specifications are described using CWL. Each individual command line tool specification can optionally interact with Docker to satisfy software dependencies. [A] The RO layer shows the structure of the RO including its content and interactions with different components in the RO and [B] the CWL layer.

the structured representation of data and metadata needed to be shared for a given workflow enactment. The structure of the CWLProv RO complies with BagIt format such that its content and completeness can be verified with any BagIt tool or library. The checksums of the payload *data/* directory are listed in the BagIt manifest-*.txt (where the asterisk sign refers to the hashing algorithms) as well as in the RO manifest as JSON-LD [78] in *metadata/manifest.json* as described below. The remaining directories include metadata of how the workflow results were created. We systematised the aggregated resources into various collections for better understanding and accessibility for a CWL workflow execution (Figure 4).

• ***data/***

data/ is the payload (collection) of all input and output files used in a given workflow enactment. Data should be labelled and identified based on a hashed checksum rather than derived from its direct occurrence during workflow execution. This use of content-addressable storage [79] simplifies identifier generation for data and helps to avoid local dependencies, e.g. hard-coded file names. However, the workflow execution engine might use customised unique identifiers for file objects. It is advised to use such identifiers to avoid redundancy and to comply with the system/platform used to run the workflow. All of the files outside *data/* should be listed in the corresponding tag manifest according to the BagIt specifications.

• ***workflow/***

CWLProv ROs should include a system-independent executable version of the workflow under the *workflow/* directory. When using CWL, this sub-directory should contain the complete workflow specification file, an input file object with parameter settings used to enact the workflow and an output file object (for understandability and not required for re-enactment) generated as a result of workflow enactment containing details of the workflow outputs such as data files produced by the workflow and other data such as string, or list outputs. These may not exactly match the files that were enacted, e.g. the absolute paths in the input job file are recommended to be replaced with relatised content-addressed paths in "*data/*" for the input and output object. The input file object should capture all the dependencies of the input data files such as indexes in case of Binary Alignment Map (BAM) format files or reference sequence files and expanding the directory to list included files.

In the case of a CWL workflow, *cwltool* facilitates aggregation of the CWL description and any referenced external descriptions (such as sub-workflows or command line tool descriptions) into a single executable file. This feature is used in our implementation (details in Section **Practical Realisation of CWLProv**) to rewrite the workflow files making them re-executable without depending on workflow or commandline descriptions outside the associated RO. Other workflow definitions or CWL executors should also apply similar features to create executable workflow objects.

• ***snapshot/***

snapshot/ comprises copies of the workflow and tool specifications files "as-is". It is recommended to use these resources just for validity of results and for understanding the workflow enactment, since these files might contain absolute paths or be host-specific, i.e. they may not necessarily be re-enacted elsewhere.

• ***metadata/***

Each CWLProv RO should contain an RO manifest file and two sub-directories *metadata/logs* and *metadata/provenance*. The RO manifest follows the structure defined for Research Object Bundles [80] but *.ro/* is instead referred as *metadata*. Further detail about the manifest file contents is documented on GitHub as *CWLProv* specification [81]. The raw log information should be made available in *metadata/logs*. This ensures inclusion of the actual commands executed including the default parameters for each step.

The retrospective provenance profiles for the workflow execution associated with the ROs require rich metadata. These profiles should exist in the *metadata/provenance*. It is recommended to make the availability of a *primary* profile mandatory and this should conform with the PROV-N format. This file describes the top-level workflow execution. As described in Level 2 (Section **Levels of Provenance and Resource Sharing**),

it is quite possible to have nested workflows. In that case, a provenance profile for all such workflows should be included in this directory. If there are additional formats of the provenance files such as JSON, XML etc, then these should be included in the said directory and a declaration of *"conformsTo"* to declare their formats in the RO manifest is mandatory. The nested workflow profile should be named such that there is a link between the respective step in the primary workflow and the nested workflow preferably using unique identifiers.

As the PROV-DM has a generalised structure, there might be some provenance aspects specific to particular workflows that are hard to capture if only using PROV-N, hence ontologies such as *wfdesc* [82] can be used to describe the abstract representation of the workflow and its steps. Use of *wfprov* [83] to capture some workflow provenance aspects is also encouraged. Alternative extensions such as ProvOne [84] can also be utilised if the Workflow Management System or workflow executor is using these extensions already.

CWLProv reuses Linked Data standards like JSON-LD, W3C PROV and ROs. A challenge with Linked Data in distributed and desktop computing is how to make identifiers that are absolute URIs and hence globally unique. For example, for *CWLProv* a workflow may be executed by an engine that does not know where its workflow provenance will be stored, published or finally integrated. To this end *CWLProv* generators should use the proposed arcp [85] URI scheme to map local file paths within the RO BagIt folder structure to absolute URIs for use within the RO manifest and associated PROV traces. Consumers of *CWLProv* ROs that do not contain an arcp-based External-Identifier should generate a temporary arcp base to safely resolve any relative URI references not present in the *CWLProv* folder. Implementations processing a *CWLProv* RO may convert arcp URIs to local *file:///* or *http://* URIs depending on how and where the *CWLProv* RO was saved, e.g. using the "arcp.py" library [86].

Retrospective Provenance Profile

As stated earlier, the provenance profile should conform to PROV-N (serialisation of PROV-DM) and may optionally contain ontologies specific to the workflow execution. In this section, we introduce few key features used in the structure of the retrospective provenance profile for a CWL workflow enactment in *CWLProv* (Table 2). The features are not tied to any platform or workflow definition approach and hence can be used to document retrospective provenance of any workflow irrespective of the workflow definition approach.

- ***wfdesc***

As PROV is a general standard, it lacks features to relate a plan (i.e. a workflow description) with sub-plans. To tackle this, we use *"wfdesc:Workflow"* for the workflow specification and *"wfdesc:hasSubProcess"* to relate individual steps to the workflow. Each step is described by *"wfdesc:Process"*. This provides the prospective view of the workflow specification without the requirement for executing it.

- ***wfprov***

This term is used for the activities and entities used in a given workflow enactment that relate to retrospective provenance. *"wfprov:WorkflowRun"* and *"wfprov:ProcessRun"* represent the workflow and individual command line tool execution components respectively. Data items are also described by *"wfprov:Artifact"*, which is a subclass of *prov:Entity*.

- ***Entity***

Entity refers to any data artefact, input configuration file, workflow or command line tool specification. Each data artefact is identified using hash values depending on the hash algorithm used to store files in *"data"*.

- ***Activity***

A workflow enactment or command line tool invocations are classified as activities that can be identified by a Universal Unique Identifier (UUID) and labelled with the absolute name of the step given in the workflow file to improve the readability.

- ***Agent***

A *"SoftwareAgent"* represents the engine used to execute the Workflow Plan. It is identified by a Unique identifier, e.g. a UUID. If a Docker container is utilised in any step, it is also documented in the profile as a *"SoftwareAgent"*. In addition, the user enacting the workflow can also be referred as an *Agent* [87] and used to store attribution details.

- ***wasAssociatedWith***

An association is represented as *wasAssociatedWith()* in PROV-N. It is used to assign a responsibility to an agent for a given activity. It relates activities such as *WorkflowRun* and *ProcessRun* to the *SoftwareAgent*, e.g. the workflow engine or Docker container.

- ***wasStartedBy***

Start is represented as *wasStartedBy* in PROV-N and is used to denote when an activity was either started by an entity (trigger) or another activity (starter). It records an activity's identifier, the starter and the time of commencement. In the case of a *WorkflowRun*, the starter is not specified as the workflow enactment is not triggered by another activity, whilst in the case of a *ProcessRun*, the corresponding *WorkflowRun* is specified as starter.

- ***Used***

In PROV-N, *used()* represents the "Usage", i.e. when an entity participates in an activity. *WorkflowRun* and *ProcessRun* both utilise this concept so that the same entity representing a data artefact can be used by *WorkflowRun* and *ProcessRun* albeit at different instances of time. To improve readability, *Prov:role* can be included as an optional parameter to *used()* to assign the name string given by users to certain inputs and outputs in a workflow description.

- ***wasGeneratedBy***

The concept of generation is represented as *wasGeneratedBy()* in PROV-N. This refers to the phenomenon where an entity is created (generated) by an activity and can only be used after its generation. It associates all output data artefacts, represented as entities with the respective activity that generated them. It is noted that the same entity can be generated by *ProcessRun* and *WorkflowRun* if it is declared as an output at both levels.

- ***wasEndedBy***

wasEndedBy() is used for the concept "end", i.e. the time when an activity is ended by another activity (ender) or by an entity (trigger). In the provenance profile, it represents the relationship between *ProcessRun* and *WorkflowRun* and the time when the *ProcessRun* is ended by the *WorkflowRun*. In addition, we also include the relationship between *WorkflowRun* and *SoftwareEngine* using this property as each workflow enactment is triggered and terminated by the engine itself.

- ***has_provenance***

As described in Section **Levels of Provenance and Resource Sharing**, in order to achieve maximum *white-box* provenance,

Table 2. *CWLProv* profile of W3C PROV, extended with Research Object Model's *wfdesc* (prospective provenance) and *wfprov* (retrospective provenance).

Level	PROV type	Subtype	Relation	Range
1	**Plan**	wfdesc:Workflow	wfdesc:hasSubProcess	wfdesc:Process
1		wfdesc:Process		
0	**Activity**	wfprov:WorkflowRun	wasAssociatedWith	wfprov:WorkflowEngine
1			↳ hadPlan	wfdesc:Workflow
0			wasStartedBy	wfprov:WorkflowEngine
0			↳ atTime	*ISO8601 timestamp*
2			wasStartedBy	wfprov:WorkflowRun
1			wasEndedBy	wfprov:WorkflowEngine
0			↳ atTime	*ISO8601 timestamp*
1		wfprov:ProcessRun	wasStartedBy	wfprov:WorkflowRun
1			↳ atTime	*ISO8601 timestamp*
1			used	wfprov:Artifact
1			↳ role	wfdesc:InputParameter
1			wasAssociatedWith	wfprov:WorkflowRun
1			↳ hadPlan	wfdesc:Process
1			wasEndedBy	wfprov:WorkflowRun
1			↳ atTime	*ISO8601 timestamp*
2		SoftwareAgent	wasAssociatedWith	wfprov:ProcessRun
2			↳ *CWLProv*:image	*docker image id*
0	**SoftwareAgent**	wfprov:WorkFlowEngine	wasStartedBy	Person *ORCID*
0			label	*cwltool --version*
1	**Entity**	wfprov:Artefact	wasGeneratedBy	wfprov:Processrun
1			↳ role	wfdesc:OutputParameter
1	**Collection**	wfprov:Artefact	hadMember	wfprov:Artefact
1		Dictionary	hadDictionaryMember	wfprov:Artefact
1			↳ pairKey	*filename*

Indentation with ↳ indicates n-ary relationships which are expressed differently depending on PROV syntax. Namespaces: http://www.w3.org/ns/prov# (default), http://purl.org/wf4ever/wfdesc# (*wfdesc*), http://purl.org/wf4ever/wfprov# (*wfprov*), https://w3id.org/cwl/prov# *CWLProv*)

the inner workings of a nested workflow should also be included in the provenance trace. If a step represents a nested workflow, a separate provenance profile is included in the RO. Moreover, in the parent workflow trace, this relationship is recorded using *has_provenance* as an attribute of the *Activity* step which refers to the profile of the nested workflow.

Practical Realisation of CWLProv

CWLProv [81] provides a format that can be adopted by any workflow executor or platform, provided that the underlying workflow definition approach is at least as declarative as CWL, i.e. it captures the necessary components described in Section **Applied Standards and Vocabularies**. In the case of CWL, as long as the conceptual constructs are common amongst the available implementations and executors, a workflow enactment can be represented in *CWLProv* format. To demonstrate the practical realisation of the proposed model we consider a Python-based reference implementation of CWL *cwltool*.

cwltool is a feature complete implementation of CWL. It provides extensive validation of CWL files as well as offering a comprehensive set of test cases to validate new modules introduced as extensions to the existing implementation. Thus it provides the ideal choice for implementing *CWLProv* for provenance support and resource aggregation. The existing classes and methods of the implementation were utilized to achieve various tasks such as packaging of the workflow and all associated tool specifications together. In addition, the existing python library *prov* [88] was used to create a provenance document instance and populate it with the required artefacts generated as the workflow enactment proceeds.

CWLProv is built as an optional module which when invoked as *"cwltool --provenance 'RO-name' workflow.cwl job.json"*, will automatically generate an RO with the given name without requiring any additional information from the user. Each input file is assigned a hash value and placed in the directory *"RO-name/data"*, making it content-addressable to avoid local dependencies (Figure 5). In order to avoid including information about attribution without consent of the user, we introduce an additional flag *" --enable-user-provenance"*. If a user provides the options *--orcid* and *--full-name*, this information will be included in the provenance profile related to user attribution. Enabling *" --enable-user-provenance"* and not providing the full name or ORCID will store details of the local machine for attribution, i.e. the details of the *agent* that enacted the workflow.

The workflow and command line tool specifications are aggregated in one file to create an executable workflow and placed in directory *RO-name/workflow*. This directory also contains transformed input job objects containing the input parameters with references to artefacts in the *RO-name/data* based on relativising the paths present in the input object. These two files are sufficient to re-enact the workflow, provided the other required artefacts are also included in the RO and comply to the *CWLProv* format. The *cwltool* control flow [89] indicates the points when the execution of the workflow and command line tools involved in the workflow enactment start, end and how the output is reported back. This information and the artefacts are captured and stored in the RO.

When the execution of a workflow begins, *CWLProv* extensions to *cwltool* generate a provenance document (using the *prov* library) which includes default namespaces for the workflow enactment *"activity"*. The attribution details as an *agent* are also added at this stage if user provenance capture is enabled, e.g. to answer "who ran the workflow?". Each step of the workflow can correspond to either a command line tool or another nested workflow referred to as a *sub-workflow* in the CWL documentation. For each nested workflow, a separate provenance profile is initialised recursively to achieve a *white-box* finer-grained provenance view as explained in Section **Lev-**

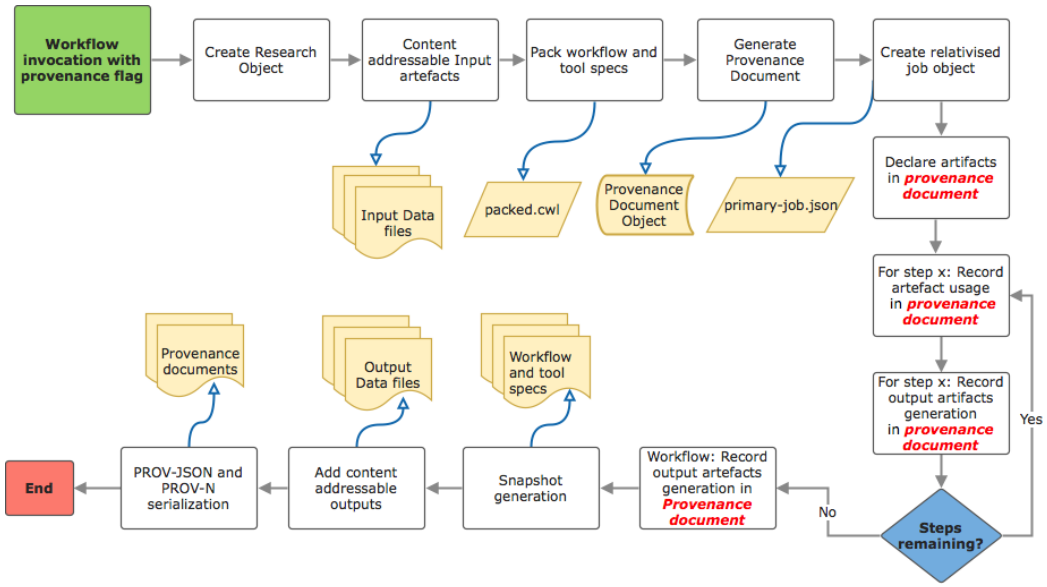

Figure 5. High level process flow representation of retrospective provenance capture

els of Provenance and Resource Sharing. This profile is continually updated throughout the nested workflow enactment. Each step is identified by a unique identifier and recorded as an *"activity"* in the parent workflow provenance profile, i.e. the *"primary profile"*. The *nested* workflow is recorded as a step in the *primary profile* using the same identifier as the "nested workflow enactment activity" identifier in the respective provenance profile. For each step in the activity, the start time and association with the workflow activity is created and stored as part of the overall provenance to answer the question "when did it happen?".

The data used as input by these steps is either provided by the user or produced as an intermediate result from the previous steps. In both cases, the *Usage* is recorded in the respective provenance profile using checksums as identifiers to answer the question "what was used?". The non-file input parameters such as strings and integers are stored "as-is" using an additional optional argument, *prov:value*. Upon completion, each step typically generates some data. The provenance profile records the generation of outputs at the step level to record "what was produced?" and "which process produced it?". Once all steps complete, the workflow outputs are collected and the generation of these outputs at the workflow level are recorded in the provenance profile. Moreover, using the checksum of these files generated by the *cwltool*, content-addressable copies are saved in the directory *"RO-name/data"*. The provenance profile refers to these files using the same checksum such that they are traceable or can be used for further analysis if required. The workflow specification, command line tool specifications and JSON job file is archived in the *"snapshot"* to preserve the actual workflow history.

This prototype implementation provides a model and guidance for workflow platforms and executors to identify their respective features that can be utilised in devising their own implementation of *CWLProv*.

CWLProv Evaluation with Bioinformatics Workflows

CWLProv supports *syntactic*, *semantic* and *pragmatic* interoperability (defined in Section **Interoperability**) of a given workflow and its associated results. We have defined a *"common data format"* for workflow sharing and publication such that any executor or workflow system with CWL support can interpret this information and make use of it. This ensures the *syntactic* interoperability between the workflow executors on different computing platforms. Similarly the *"content"* of the shared aggregation artefact as a workflow-centric RO is unambiguously defined, thus ensuring uniform representation of the workflow and its associated results across different platforms and executors hence supporting *semantic* interoperability. With level 3 provenance satisfied providing domain-specific information along with level 0-2 provenance tracking, we posit that CWLProv accomplishes *pragmatic* interoperability by providing unambiguous information about the *"context"*, *"application"* and *"use"* of the shared/published workflow-centric ROs.

To demonstrate the interoperability and portability of the proposed solution, we evaluate *CWLProv* and its reference implementation using open source bioinformatics workflows available on GitHub from different research initiatives and from different developers.

RNA-seq Analysis Workflow

RNA sequencing (RNA-seq) data generated by Next Generation Sequencing (NGS) platforms is comprised of short sequence reads that can be aligned to a reference genome, where the alignment results form the basis of various analyses such as quantitating transcript expression; identifying novel splice junctions and isoforms and differential gene expression [91]. RNA-seq experiments can link phenotype to gene expression and are widely applied in multi-centric cancer studies [25]. Computational analysis of RNA-seq data is performed by different techniques depending on the research goals and the organism under study [92]. The workflow [93] included in this case study has been defined in CWL by one of the teams [94] participating in NIH Data Commons initiative [95], an large research infrastructure program aiming to make digital objects (such as data generated during biomedical research and software/tools required to utilise such data) shareable and accessible and hence aligned with the FAIR principles [96].

This workflow (Figure 6), designed for the pilot phase of the NIH Data Commons initiative [97], adapts the approach and parameter settings of Trans-Omics for precision Medicine (TOPMed) [98]. The RNA-seq pipeline originated from the Broad Institute [99]. There are in total five steps in the work-

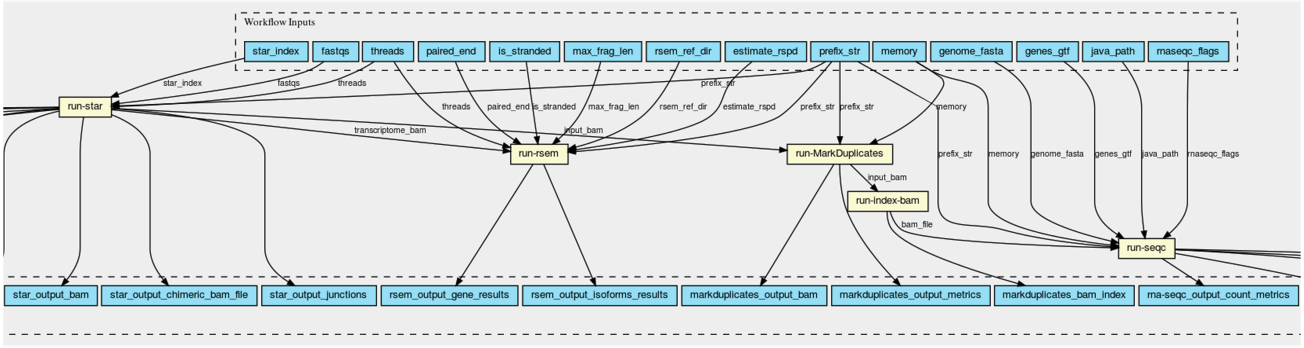

Figure 6. Snippet of RNA-seq workflow generated by CWL viewer [90].

flow starting from: 1. Read alignment using STAR [100] which produces aligned BAM files including the Genome BAM and Transcriptome BAM. 2. The Genome BAM file is processed using Picard MarkDuplicates [101] producing an updated BAM file containing information on duplicate reads (such reads can indicate biased interpretation). 3. SAMtools index [102] is then employed to generate an index for the BAM file, in preparation for the next step. 4. The indexed BAM file is processed further with RNA-SeQC [103] which takes the BAM file, human genome reference sequence and Gene Transfer Format (GTF) file as inputs to generate transcriptome-level expression quantifications and standard quality control metrics. 5. In parallel with transcript quantification, isoform expression levels are quantified by RSEM [104]. This step depends only on the output of the STAR tool, and additional RSEM reference sequences.

For testing and analysis, the workflow author provided example data created by down-sampling the read files of a TOPMed public access data [105]. Chromosome 12 was extracted from the *Homo Sapien Assembly 38* reference sequence and provided by the workflow authors. The required GTF and RSEM reference data files are also provided. The workflow is well-documented with a detailed set of instructions of the steps performed to down-sample the data are also provided for transparency. The availability of example input data, use of containerization for underlying software and detailed documentation are important factors in choosing this specific CWL workflow for CWLProv evaluation.

Alignment Workflow

Alignment is an essential step in variant discovery workflows and considered an obligatory *pre-processing* stage according to Best Practices by the Broad Institute [62]. The purpose of this stage is to filter low-quality reads before variant calling or other interpretative steps [106]. The workflow for alignment is designed to operate on raw sequence data to produce analysis-ready BAM files as the final output. The typical steps followed include file format conversions, aligning the read files to the reference genome sequence, and sorting the resulting files. The CWL alignment workflow [107] included in this evaluation (Figure 7) is designed by Data Biosphere [108]. It adapts the alignment pipeline [109] originally written by Hyun Min Kang and Adrian Tan at Abecasis Lab [110], The University of Michigan. This workflow is also part of NIH Data Commons initiative (as RNA-seq Analysis Workflow) and comprises of four stages. First step, "Pre-align" accepts a Compressed Alignment Map (CRAM) file (a compressed format for BAM files developed by European Bioinformatics Institute (EBI) [111]) and human genome reference sequence as input and using underlying software utilities of SAMtools such as view, sort and fixmate returns a list of fastq files which can be used as input for the next step. CRAM file "Align" also accepts the human reference genome as input and uses BWA-mem [112] to generate aligned reads as BAM files which are sorted in the next step "SAMtool sort". Finally these sorted alignment files are merged to produce single sorted BAM file using SAMtools merge in "Post-align" step. The authors provide an example CRAM file, *Homo Sapien Assembly 38* reference genome along with its index files to be used as inputs for testing and analysis of the workflow.

Somatic Variant Calling Workflow

Variant discovery analysis for high-throughput sequencing data is a widely used bioinformatics technique, focused on finding genetic associations with diseases, identifying somatic mutations in cancer and characterizing heterogeneous cell populations [113]. The *pre-processing* explained for the Alignment workflow is part of any variant calling workflow as reads are classified and ordered as part of the variant discovery process. Numerous variant calling algorithms have been developed depending on the input data characteristics and the specific application area [106]. Somatic variant calling workflows are designed to identify somatic (non-inherited) variants in a sample - generally a cancer sample - by comparing the set of variants present in a sequenced tumour genome to a non-tumour genome from the same host [114]. The set of tumour variants is a super-set of the set of host variants, and somatic mutations can be identified through various algorithmic approaches to subtracting host familial variants. Each somatic variant calling workflow typically consists of three stages: preprocessing; variant evaluation and post-filtering.

The somatic variant calling workflow (Figure 8) included in this case study is designed by Blue Collar Bioinformatics (bcbio) [115], a community-driven initiative to develop best-practice pipelines for variant calling, RNA-seq and small RNA analysis workflows. According to the documentation, the goal of this project is to facilitate the automated analysis of high throughput data by making the resources *quantifiable*, *analyzable*, *scalable*, *accessible* and *reproducible*. All the underlying tools are containerized facilitating software use in the workflow. The somatic variant calling workflow defined in CWL is available on GitHub [116] and equipped with a well defined test dataset.

Evaluation Activity

This section describes the evaluation of cross-executor and cross-platform interoperability of CWLProv. To test cross-executor interoperability, two CWL executors *cwltool* and *toil-cwl-runner* were selected. *toil-cwl-runner* is an open source Python workflow engine supporting robust cross-platform workflow execution on Cloud and High Performance Comput-

ing (HPC) environments [118]. The two operating system platforms utilised in this analysis were MacOS and Ubuntu Linux. For the Linux OS, a 16-core Linux instance with 64GB RAM was launched on the Australian National eResearch Collaboration Tools and Resources (NeCTAR) research cloud [119]. To cater for the storage requirements, a 1000GB persistent volume was attached to this instance. For MacOS, a local system with 16GB RAM, 250GB storage and 2.8 GHz Intel Core i7 processor was used. These platforms were selected to cater for the required storage and compute resources of the workflows described above. The reference genome provided with Alignment Workflow was not down-sampled and hence this workflow required most resources among the three evaluated.

It is worth mentioning that this evaluation does not include details of the installation process for *cwltool*, *toil-cwl-runner* and *Docker* on systems described above. To create *CWLProv* ROs during workflow execution, it is necessary to use the CWL reference runner (*cwltool*) until this practice spreads to other CWL implementations. Moreover, it is assumed that the software container (Docker) should also be installed on the system to use the workflow definitions aggregated in a given *CWLProv* RO.

In addition, the resource requirements (discussed in Section **Discussion and Future Directions**) should also be satisfied by choosing a system with enough compute and storage resources for successful enactment. The systems used in this case study should be a reference when selecting a system as inadequate compute and storage resources such as insufficient RAM or number of cores will hinder the successful re-enactment of workflows using these ROs. The hardware requirements may also vary if a different dataset is used as input to re-enact the workflow using the methods aggregated in the RO. In that case, the end user must ensure availability of adequate compute and storage resources by choosing a system that meets the required specifications [120].

Since the CWLProv implementation is demonstrated for one of the executors (*cwltool*), currently a CWLProv RO for any workflow can only be produced using *cwltool*. Hence, in this activity the workflows are initially enacted using just *cwltool* (Table 3). The outline of the steps performed to analyse *CWLProv* for each case study is as follows.

I) The workflow was enacted using *cwltool* to produce a RO on a MacOS computer.

1) The resulting RO and aggregated resources were used to re-enact the workflow using *toil-cwl-runner* on the same MacOS computer;
2) The RO produced in step I was transferred to the cloud-based Linux instance used in this activity;
3) On the cloud-based Linux environment and only utilizing the resources aggregated in the RO, the workflow was re-enacted using *cwltool* and *toil-cwl-runner*.

II) The workflow was enacted using *cwltool* to produce a RO on Linux.

1) The resulting RO and aggregated resource were utilized to re-enact the workflow using *toil-cwl-runner* on the same cloud-based Linux instance;
2) The RO produced in step II was transferred to the MacOS computer used in this activity;
3) On the MacOS computer and only utilizing the resources aggregated in the RO, the workflow was re-enacted using *cwltool* and *toil-cwl-runner*.

Evaluation Results

The following steps were taken to produce ROs which were then used to re-enact the workflows outlined in Table 3, without any further changes required. This demonstration illustrates the syntactic, semantic and pragmatic portability of the workflows across different systems. It shows that **both CWL executors were able to *exchange*, *comprehend* and *use* the information represented as CWLProv ROs**.

Table 3. *CWLProv* evaluation summary and status for the 3 bioinformatics case studies.

Enact-produce RO with	Re-enact using RO with	Status
cwltool on MacOS	*toil-cwl-runner* on MacOS	✓
	cwltool on Linux	✓
	toil-cwl-runner on Linux	✓
cwltool on Linux	*toil-cwl-runner* on Linux	✓
	cwltool on MacOS	✓
	toil-cwl-runner on MacOS	✓

• ***CWLProv and Interoperability***
CWL already builds on technologies such as JSON-LD for data modeling and Docker to support portability of the run-time environments. The portability and interoperability as basic principles of the underlying workflow definition approach for any workflow-centric analysis implies that the analysis should also be portable and interoperable. However, the workflow definition/specification alone is insufficient when dealing with commandline tool specifications, data, and input configuration files used in the analysis if these are not readily available.

CWLProv ensures availability of these resources for a given analysis conforming to the framework defined in Section **CWLProv and utilised standards**. The input configurations are saved as *primary-job.json* in directory *workflow/* and refer to the

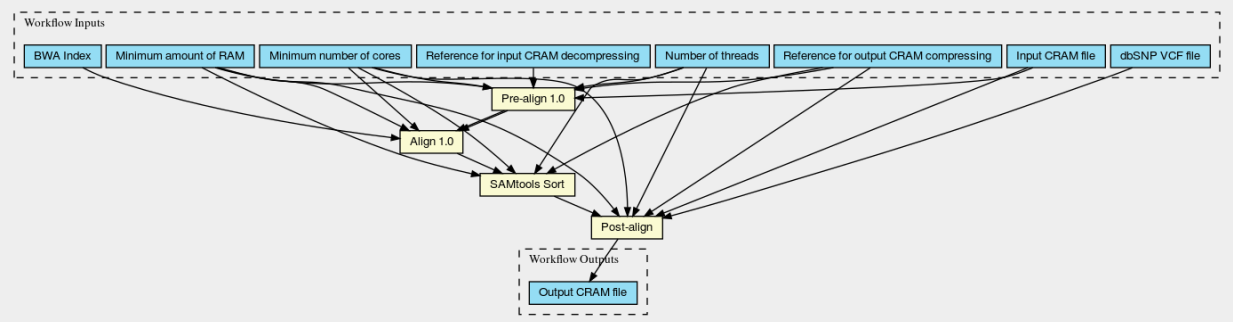

Figure 7. Alignment workflow representation generated by CWL viewer.

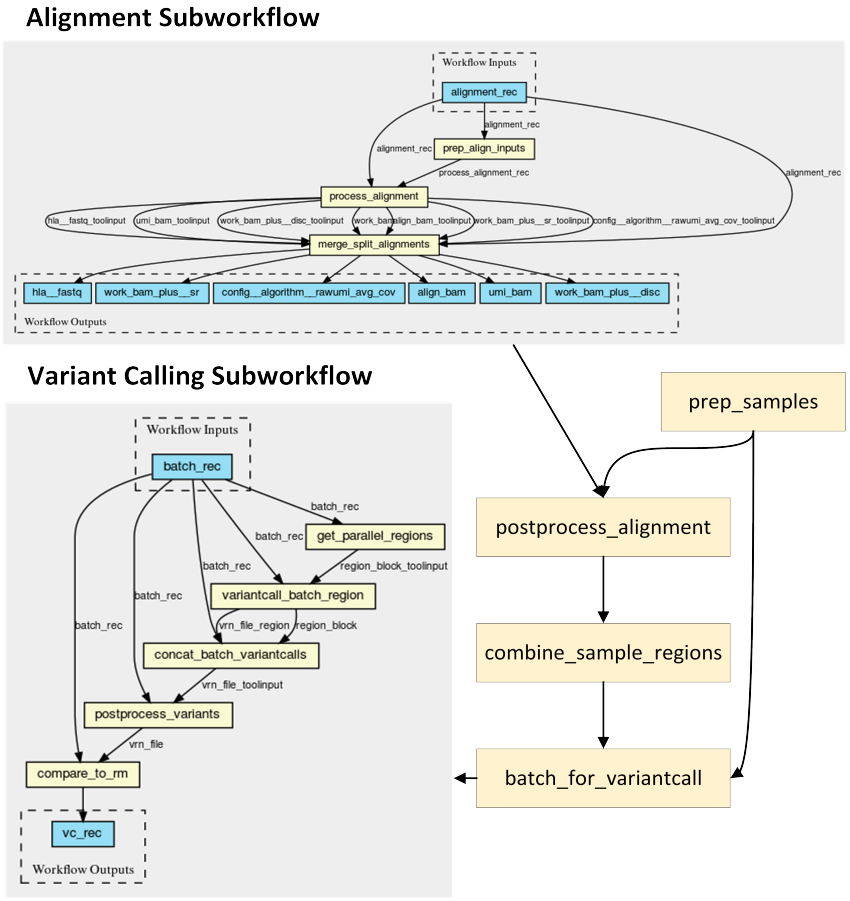

Figure 8. Visual representation of the bcbio somatic variant calling workflow (Adapted from [117] and the subworkflow images are generated by CWL viewer.

input data contained in the payload *data/* directory of the given RO. In this way, availability of data aggregated with the analysis is made possible. Existing features of *cwltool* are used to generate the CWL workflow specification file containing all of the commandline tool specifications referred to in the workflow specification and placed in the same *workflow/* directory.

One might argue that copying a directory tree might serve the same purpose but in that case we again will be relying on users to put substantial amount of effort on top of the actual analysis, i.e. they would have to carefully structure their directories to be aligned with the workflow creators. Instead CWL encourages researchers to utilise container technologies such as Docker, Singularity, Debian (Med) or Bioconda packages to ensure availability of underlying tools as recommended by numerous studies [13, 6, 52, 56, 57, 121]. This practice facilitates the preservation of methods utilised in data-intensive scientific workflows and enables verification of the published claims without requiring the end-user to do any manual installation and configuration. Examples of tools available via Docker containers used here are the alignment tool (BWA mem) used in the Alignment workflow and STAR aligner used in RNA-seq workflow.

• ***Evaluating Provenance Profile***

The retrospective provenance profile generated as part of CWLProv for each workflow enactment can be examined and queried to extract the required subset of information. *Provenance Analytics* is a separate domain and a next step after provenance collection in the provenance life cycle [122]. Often provenance data is queried using specialized query languages such as SQL SPARQL or TriQL depending on the storage mechanism used. Query operations can combine information from prospective and retrospective provenance to understand computational experiments better.

The focus of this paper is not in-depth provenance analytics but we have demonstrated the application of the provenance profile generated as part of CWLProv. We have developed a commandline tool and Python API *"cwlprov-py"* Soiland-Reyes and Khan [123] for CWLProv RO analytics to interpret the captured retrospective provenance of CWL workflow enactment. This API currently supports the following use-cases.

Given a CWLProv RO:

- **Workflow Runs**
  As each RO can contain more than one *workflow run* if sub-workflows are utilized to group related tasks into one workflow. In that case, the provenance traces are stored in separate files for each workflow run. *cwlprov-py* identifies the workflow enactments including the sub-workflows *(if any)* and returns the workflow identifiers annotated with the step names. The user can select the required trace and explore particular traces in detail.
- **Attribution**
  Each RO is assumed to be associated with a single enactment of the primary workflow and hence assumed to be enacted by one person. As discussed previously, *CWLProv* provides additional flags to enable user provenance capture. A user can provide their name and ORCID details that can be stored as part of a RO. *cwlprov-py* displays attribution details of the researcher responsible for the enactment *(if enabled)* and the versions of the workflow executor utilised in the analysis.
- **Input/Output of a Process**
  Provenance traces contain associations between the steps/workflows with the data they used or generated. A

Table 4. Run-time comparison for the workflow enactments done cross-executor and cross-platform

Workflow	Linux			MacOS		
	cwltool		toil-cwl-runner	cwltool		toil-cwl-runner
	With Prov	W/O Prov	W/O Prov	With Prov	W/O Prov	W/O Prov
RNA-Seq Analysis Workflow	4m30.289s	4m0.139s	3m46.817s	3m33.306s	3m41.166s	3m30.406s
Alignment Workflow	28m23.792s	24m12.404s	15m3.539s	–	162m35.111s	146m27.592s
Somatic Variant Calling Workflow	21m25.868s	19m27.519s	7m10.470s	17m26.722s	17m0.227s	**

** This could not be tested because of a known bug: https://github.com/common-workflow-language/cwltool/issues/328
– This could not be tested because of the insufficient hardware resources on the MacOS

user interested in a particular step or a workflow enactment can identify the inputs used and outputs produced linked explicitly to that process using *cwlprov-py*.

- **Partial Re-runs**
  Re-running or re-using only desired parts of a given workflow has been emphasized [25] as important to evaluate the workflow process or validate the published results associated without necessarily re-enacting the workflow as a whole. *cwlprov-py* uses the identifier of the step/workflow to be re-run, parses the provenance trace to identify the inputs required and ultimately creates a JSON input object with the associated input parameters. This input object can then be used for partial re-runs of the desired step/workflow making segmented analysis possible without unnecessary computations.

• ***Temporal and Spatial Overhead with Provenance***

Table 4 shows the run-times for the three workflow enactments using cwltool and toil-cwl-runner on Linux and MacOS with and without enabling provenance capture as described in the evaluation activity section. These workflows were enacted at least once before this time calculation, hence the timing does not include the time for Docker images to be downloaded. On a new system, when re-running these workflows for the first time, the Docker images will be downloaded and may take significantly longer than the time specified here especially in case of the Somatic Variant Calling workflow because of the image size.

Run-time and storage overheads are important for provenance-enabled computational experiments. The choice of different operating systems and provenance capture mechanisms such as operating-system level, application-level or workflow-level as well as I/O workload, interception mechanism and fine-grained information capture are key for provenance [124, 125].

In our case study, significant time difference can be seen for the alignment workflow that used the most voluminous dataset, hence producing a sizable RO as well. This was due to the RO-generation where data was aggregated within the RO. The difference between the provenance-enabled enactment versus the enactment without provenance is barely noticeable for the other two workflow enactments with the smaller datasets. The discussion about handling the big '-omics' data such as human genome reference sequence, its index files and other database files (e.g. dbsnp) in Section **Discussion and Future Directions** provides a possible solution to avoid such overheads.

In addition, noticeable time difference between the cwltool and toil-cwl-runner enactments is because of the default parallel versus serial job execution in case of *toil-cwl-runner* and *cwltool* respectively. The "scatter" operation in CWL when applied to one or more input parameters of a workflow step or a sub-workflow, supports parallel execution of the associated processes. Parallelism is also available without "scatter" when separate processes have all their inputs ready. If sufficient compute resources are available, these jobs will be enacted concurrently otherwise they are queued for subsequent execution. Compute intensive steps of a workflow can benefit from scatter features for parallel execution by reducing the overall runtime. Both Alignment and Somatic Variant Calling workflows utilise the scatter feature to enable higher degrees of parallel job execution in case of *toil-cwl-runner* which explains the time difference for the cross-executor of these two workflows. The difference is negligible for RNA-Seq workflow which is comprised of serial jobs with comparatively small test data.

• ***Output Comparison Across Enactments***

We compared the workflow outputs after each enactment to observe the concordance and/or discordance (if any) for the workflow enactment results produced across the platforms and across the executors. As CWLProv RO refers to the data with hashed checksums, these checksums are utilised for the result comparison. It is worth-mentioning that the comparison was made between the output files generated by the different enactments against a single *"truth-set"* output file available and checksum in the respective Git repositories.

The checksum of the output data generated cross-platform and cross-executor comparison data as a result of the initial enactments and re-runs using the CWL ROs to elicit the concordance in all but one cases. The "correctness" as well as agreement of these outputs given different execution environments (e.g. platform and executor) hold true except for Alignment workflow. Alignment workflow produced varying outputs after every execution even with the same executor and platform. The output of the alignment algorithm, "BWA mem" used in this workflow was non-deterministic as it depended on the *number of threads* $-t$ and the *seed length* $-K$ which affected the output produced. While the seed length in this case was set to a constant value, the number of threads varied depending on the availability of hardware resources at run-time, thereby resulting in varying output for the same input files.

Discussion and Future Directions

This section discusses the current and future work with reference to enriched provenance capture and smart resource aggregation, and enhancements to both the CWLProv standard and implementation.

• ***Compute and Storage Resources***

The *CWLProv* format encapsulates the data and workflow definitions involved in a given workflow enactment along with its retrospective provenance trace. CWL as a standard provides constructs to declare basic hardware resource requirements such as minimum and maximum cores, RAM and reserved file system storage required for a particular workflow enactment. The workflow authors can provide this information in the *"requirements"* section as *"ResourceRequirement"*. These requirements can be declared at workflow or individual step level, to help platforms/executors to allocate the required resources. This information indirectly stores some aspects of

prospective view of provenance with respect to hardware requirements of the underlying system used to enact a workflow. Currently this information is only available if declared as part of workflow specification. In future, we plan to include these requirements as part of provenance for a given workflow such that all such information is gathered in one space and users are not required to inspect multiple sources to extract this information. This information can then be used as a pre-condition for potential successful enactment of a given workflow.

As *CWLProv* is focused on retrospective provenance capture of workflow enactment, we plan to include provenance information about the compute and storage resources utilised in a given enactment to fulfill *R19*. We believe that documenting these resources will allow users to analyse their environment and resource allocations before execution, as opposed to trial and error methods that may result in multiple failed enactments of a given workflow. Despite being an important factor, it is surprising to see that most of existing provenance standards lack dedicated constructs to represent the underlying hardware resource usage information as part of prospective or retrospective provenance. In the case of complex workflows using distributed resources, where each step could be executed on a different node/server, including all this information in a single *PROV* profile will clutter the profile and render it potentially incomprehensible. Therefore, we plan to add a separate *Usage Record* document in the RO conforming to GFD.204 [126] to describe level 1 (and potentially level 2) resource usage in a common format independent on actual execution environment. Capturing such resource usage records require a tighter integration with the execution platform, and so we consider this future work better suited for a cloud-based CWL engine like *Toil* or *Arvados*, as the reference implementation *cwltool* does not exercise fine-grained control of its task execution. Detailed raw log files can also be provided as level 0 provenance, as we have demonstrated with cwltool, but these will by their nature be custom per execution platform and thus should be considered unstructured.

• ***Provenance Profile Augmented with Domain Knowledge***
CWLProv benefits from existing best practices proposed by numerous studies (Table 1) and includes defined standards for workflow representation, resource aggregation and provenance tracking (Section **Applied Standards and Vocabularies**). We posit that the principle of following well-defined data and metadata standards enables explicit data sharing and reuse. In order to include rich metadata for bioinformaticians to produce specialised ROs for bioinformatics to achieve CWLProv level 3 as defined in Section **Levels of Provenance and Resource Sharing**, we are investigating re-use of concepts from the BioCompute Object (BCO) project [67]. This domain-specific information is not necessary for computation and execution but for understandability of the shared resources. We encourage workflow authors to include such metadata and external identifiers for data and underlying tools, e.g. EDAM identifiers. These annotations will be extracted and represented in the retrospective provenance profile in *CWLProv*. Domain-specific information is essential in determining the nature of inputs, outputs and context of the processes linked to a given workflow enactment [63]. This information can be captured in the RO if and only if the workflow author adds it in the workflow definition, thus achieving CWLProv level 3 depends on the individual workflows.

• ***Big -omics Data***
While aggregating all resources as one download-able object improves reproducibility, the size of the resulting RO is an important factor in practice. On one hand, completeness of the resources contributes towards minimizing the *workflow decay* phenomenon by least dependence on availability of third party resources. On the other hand, the nature of -omics data sizes can result in hard-to-manage workflow-centric ROs also leading to the spatial and temporal overheads as discussed in evaluation.

One solution is archiving the big datasets in online repositories or data stores and including persistent identifiers and checksums in the RO instead of the actual data files, as previously demonstrated with BDBags [74]. While CWL executors like *toil-cwl-runner* can be configured to deposit data in a shared repository, the *cwltool* reference implementation explored in this study can only write to the local file system. External references raise the risk of unavailability of data at a later time. Therefore we recommend including the data in the RO if sufficient network and storage resources are available. Future work may explore post-processing CWLProv ROs to replace large data files with references to stable data repositories, producing a slimmer RO for transfer where individual data items can be retrieved on demand, as well as reducing data duplication across multiple related ROs.

• ***Customized CWLProv RO with Selected Jobs Provenance***
Workflows may have intermediate steps in which data are large, but not particularly interesting, such as converting a tab-separated file to a comma-separated file. Provenance data from such steps effectively double the storage cost with little information gain, as such data can be reliably recreated by re-applying the transformation step. Future *CWLProv* work could add options to ignore capturing outputs of specified *shim* steps, or to ignore files over a particular file size. Shim refers to an adapter step required to resolve the format incompatibility issues between two workflow tasks [54]. Shim step employs tools to convert the output of the previous step into an acceptable format for the next step in a workflow. For example in our case study RNA-seq workflow, RNA-SeQC required an indexed BAM file whereas the output of STAR or Picard MarkDuplicates only comprises of the BAM file. Hence, SAMtools index was utilised afterwards to make the aligned reads analysis ready for RNA-SeQC. Similarly a scientist or a workflow system may only capture provenance at a particular provenance level (see Section **Levels of Provenance and Resource Sharing**). It is envisioned that such partial provenance capture can be indicated in the RO manifest as a variant of the CWLProv profile identifier to give the end-user clear indication of what to expect in terms of completeness.

• ***Enforcement of Best Practices – An Open Problem***
Recommendations and best practices from the scientific community are proposed frequently, to guide researchers to design their computational experiments in such a way as to make their research reproducible and verifiable. Not only the best practices for workflow design, but also for resource declaration, software packaging and configuration management are put forward [121] to avoid dependence on local installations and manual processes of dependency management. The term *"Better Software, Better Research"* [127] can also be well-applied on and adapted for the workflow design process.

Declarative approaches to workflow definition such as CWL facilitate and encourage users to explicitly declare everything in a workflow, improving white-box view of the retrospective as well as prospective provenance. Such workflows should provide insights of the complete process followed, to produce a data artefact resolving the black-boxness often associated with the workflow provenance. However, it is entirely up to researchers to leverage these approaches to produce well-defined workflows with explicit details facilitating enriched capture of the provenance trace at the appropriate level, and this can require considerable effort and consistency on the workflow de-

signer's behalf. For instance, the alignment workflow used in this case study embeds bash scripts into the CWL tool definition, therefore requiring another layer needed to be penetrated for provenance information extraction. Despite using CWL for the workflow definition and CWLProv for provenance capture, the provenance trace will be missing critical information making it coarse-grained, and the raw logs capturing the enactment will also not be as informative.

The three criteria defined by Cohen-Boulakia et al. [25] to be followed by workflow designers are: modularized specifications, unified representation and workflow annotations. CWL facilitates a modular structure to workflow definitions by coupling similar steps to *subworkflows*; and, as an interoperable standard, CWL provides a common platform moving towards resolution of the heterogeneity of the workflow specification languages. In addition, users can add domain-specific annotations to data and workflows to enhance understanding of the shared specification. All these features can be utilised to design better workflows and maximize the information declaration resulting in semantically-rich and provenance-complete CWLProv ROs.

Workflow-centric initiatives similar to *software carpentry* [128] and *code is science* [129] are one possible way to organize training and create awareness around best practices. Community-driven efforts to further consolidate the understanding of requirements to make a given workflow explicit, understandable and reproducible should be made. Not only awareness about the workflow design is needed, but also the availability of the associated resources should be emphasized e.g. software as containers or software packages, big datasets in public repositories and pre-processing/post-processing as part of workflow. Without putting proposed best practices into actual practice, complete sharing of a workflow-centric computational analysis is likely to remain challenging.

Conclusion

The comprehensive sharing of computational experiments employed to create a given data artefact alongside their provenance is recognized as an essential factor to establish trust and ensure reproducibility of the published studies. Shared resources are sometimes rendered ineffective due to limited annotations, heterogeneity of platforms, unavailability of software and limited access to data. To this context, the contributions of this study are four-fold. First, we have provided a comprehensive understanding of the recommendations and expectations from the community regarding workflow design and resource sharing. Second, we define a hierarchical framework offering *"Levels of Provenance and Resource Sharing"*. Each layer of this framework addresses specific provenance recommendations and supports the capture of rich provenance information, with the topmost layer enabling the sharing of comprehensive and executable workflows utilising retrospective provenance.

Third, we leverage community-driven, domain-neutral, platform-independent and open-source standards to define **"CWLProv"** - a format for the methodical representation of provenance supporting workflow enactment, capturing the associated artefacts, aggregating resources specific to the given enactment and associated workflow configuration settings. Finally, to demonstrate the applicability of *CWLProv*, we extend an existing workflow executor (cwltool) to provide a reference implementation that generates metadata and provenance-rich interoperable workflow-centric ROs, aggregating and preserving data and methods to support the coherent sharing of computational analyses and experiments. We also evaluated the interoperability and completeness of the *CWLProv* ROs generated for widely-used bioinformatics workflows and discussed the open issues and the potential solutions.

With any published scientific research, statements such as *"Methods and data are available upon request"* should no longer be acceptable in a modern open-science-driven research community. Considering on one hand the collaborative nature and emerging openness of bioinformatics research and on the other hand the heterogeneity of workflow design approaches, it is essential to share structured representation of the data and methods utilised in any scientific study to achieve interoperable solutions facilitating reproducibility of science. These goals can only be achieved by encouraging and following well-established and agreed upon best practices for workflow design, software environment deployment, complete provenance documentation and open access to all the artefacts. Hence, the usability of any *CWLProv* Research Object directly relies on the choice of practices followed to define workflows and to integrate related resources. We conclude this study by emphasizing that any given *CWLProv* RO and the retrospective provenance supplied with it can be employed effectively to support the research interoperability, given that it is augmented with the right resources that aid its comprehension and completeness.

Availability of source code and requirements

CWLProv is implemented as part of the CWL reference implementation *cwltool*:

- Project name: cwltool (RRID:SCR_015528)
- Project home page: https://github.com/common-workflow-language/cwltool
- Version: 1.0.20181012180214 [10]
- Operating system(s): Platform independent
- Programming language: Python 3.5 or later (RRID:SCR_008394)
- Other requirements: Docker (RRID:SCR_016445) recommended
- License: Apache License, Version 2.0

The *CWLProv profile* documents the use of W3C PROV in a Research Object to capture a CWL workflow run:

- Project name: CWLProv profile
- Project home page: https://w3id.org/cwl/prov
- Version: 0.6.0 [81]
- Operating system(s): Platform independent
- License: Apache License, Version 2.0

The *CWLProv Python Tool* can be used to explore CWLProv research objects on the command line:

- Project name: CWLProv Python Tool (*cwlprov-py*)
- Project home page: https://github.com/common-workflow-language/cwlprov-py
- Version: 0.1.1 [123]
- Operating system(s): Platform independent
- Programming language: Python 3.5 or later (RRID:SCR_008394)
- License: Apache License, Version 2.0

Availability of supporting data and materials

Research Objects of CWL workflow executions available at https://www.dropbox.com/sh/zkukezlflirospi/AAAABYjXeoVCGEnkyeHQyA-ba?dl=0 (to be deposited to Mendeley Data or GigaDb after peer review).

Declarations

List of abbreviations

If abbreviations are used in the text they should be defined in the text at first use, and a list of abbreviations should be provided in alphabetical order.

Ethical Approval (optional)

Not applicable.

Consent for publication

Not applicable.

Competing Interests

SSR and MRC are members of the leadership team for Common Workflow Language at the Software Freedom Conservancy.

Funding

FZK funded by Melbourne International Research Scholarship (MIRS) and Melbourne International Fee Remission Scholarship (MIFRS).

SSR funded by BioExcel CoE, a project funded by the European Commission Horizon 2020 Framework Programme under contract H2020-EINFRA-2015-1-675728.

Author's Contributions

Conceptualization: FZK, SSR, MRC. Data curation: FZK. Formal analysis: FZK. Funding acquisition: FZK, CAG. Investigation: FZK. Methodology: FZK, SSR. Project administration: FZK, SSR. Resources: ROS, AL. Software: FZK, SSR, MRC. Supervision: MRC, ROS, AL, CAG. Validation: FZK, SSR. Writing - original draft: FZK. Writing - review & editing: FZK, SSR, ROS, AL, MRC.

Acknowledgements

An earlier version of this article [130] was submitted for consideration at International Provenance and Annotation Workshop (IPAW) 2018. We would like to thank the IPAW reviewers for their constructive comments.

We would like to thank the Common Workflow Language community, and in particular Peter Amstutz, for their continuing support and feedback. We would also like to thank Brad Chapman, Christopher Ball and Lon Blauvelt for the three workflows used in the evaluation and their prompt replies whenever asked any question.

We are grateful for partial travel support from Open Bioinformatics Foundation (OBF) Travel Fellowship Program [131] to Farah Zaib Khan for attending the Bioinformatics Open Source Conference (BOSC) 2017 and 2018 Codefests subsidizing this collaborative effort.

We would like to thank Carole Anne Goble for continued support, mentoring and promotion of Common Workflow Language.

References

1. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al. Big data: astronomical or genomical? PLoS Biol 2015 jul; 13(7):e1002195, 10.1371/journal.pbio.1002195.
2. Atkinson M, Gesing S, Montagnat J, Taylor I. Scientific workflows: Past, present and future. Future Generation Computer Systems 2017 oct; 75:216–227, 10.1016/j.future.2017.05.041.
3. Spjuth O, Bongcam-Rudloff E, Hernández GC, Forer L, Giovacchini M, Guimera RV, et al. Experiences with workflows for automating data-intensive bioinformatics. Biology Direct 2015 aug; 10(1), 10.1186/s13062-015-0071-8.
4. Cuevas-Vicenttín V, Dey S, Köhler S, Riddle S, Ludäscher B. Scientific workflows and provenance: introduction and research opportunities. Datenbank Spektrum 2012 nov; 12(3):193–203, 10.1007/s13222-012-0100-z.
5. Common Workflow Language project, Existing Workflow Systems; 2018. https://s.apache.org/existing-workflow-systems, accessed 12 Sep 2018.
6. Kanwal S, Khan FZ, Lonie A, Sinnott RO. Investigating reproducibility and tracking provenance - A genomic workflow case study. BMC Bioinformatics 2017 jul; 18(1):337, 10.1186/s12859-017-1747-0.
7. Wolstencroft K, Haines R, Fellows D, Williams A, Withers D, Owen S, et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Res 2013 jul; 41(Web Server issue):W557–61, 10.1093/nar/gkt328.
8. Möller S, Prescott SW, Wirzenius L, Reinholdtsen P, Chapman B, Prins P, 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 2017 nov; 2(3):232–244, 10.1007/s41019-017-0050-4.
9. Alterovitz G, Dean DA, Goble C, Crusoe MR, Soiland-Reyes S, Bell A, et al. Enabling Precision Medicine via standard communication of NGS provenance, analysis, and results 2017 sep;, 10.1101/191783.
10. Amstutz P, Crusoe MR, Khan FZ, Soiland-Reyes S, Singh M, kumar K, et al., common-workflow-language/cwltool: 1.0.20181012180214. Zenodo ; 2018, 10.5281/zenodo.1471589.
11. Amstutz P, Crusoe MR, Nebojša Tijanić, Chapman B, Chilton J, Heuer M, et al., Common Workflow Language, v1.0. Figshare ; 2016, 10.6084/m9.figshare.3115156.v2.
12. Ivie P, Thain D. Reproducibility in Scientific Computing. ACM Comput Surv 2018 Jul; 51(3):63:1–63:36, 10.1145/3186266.
13. Belhajjame K, Zhao J, Garijo D, Gamble M, Hettne K, Palma R, et al. Using a suite of ontologies for preserving workflow-centric research objects. Web Semantics: Science, Services and Agents on the World Wide Web 2015 may; 32(0):16–42, 10.1016/j.websem.2015.01.003.
14. Network Working Group, draft-kunze-bagit-17 - The BagIt File Packaging Format (V1.0); 2017. https://tools.ietf.org/html/draft-kunze-bagit-17, accessed 22 Sep 2018.
15. Missier P, Belhajjame K, Cheney J. The W3C PROV family of specifications for modelling provenance metadata. In: Proceedings of the 16th International Conference on Extending Database Technology - EDBT '13 ACM Press; 2013. , https://doi.org/10.1145/2452376.2452478, 10.1145/2452376.2452478.
16. Hettne KM, Dharuri H, Zhao J, Wolstencroft K, Belhajjame K, Soiland-Reyes S, et al. Structuring research methods and data with the research object model: genomics workflows as a case study. J Biomed Semantics 2014 sep; 5(1):41,

10.1186/2041-1480-5-41.
17. Belhajjame K, Corcho O, Garijo D, Zhao J, Missier P, Newman D, et al. Workflow-centric research objects: First class citizens in scholarly discourse. In: Proceedings of the 2nd Workshop on Semantic Publishing (SePublica 2012), vol. 903 of CEUR Workshop Proceedings; 2012. p. 1–12. http://ceur-ws.org/Vol-903/paper-01.pdf.
18. Herschel M, Diestelkämper R, Ben Lahmar H. A survey on provenance: What for? What form? What from? VLDB J 2017 dec;26(6):881–906.
19. PROV-DM: The PROV Data Model;. https://www.w3.org/TR/prov-dm/, accessed 22 Sep 2018.
20. Clifford B, Foster I, Voeckler JS, Wilde M, Zhao Y. Tracking provenance in a virtual data grid. Concurrency and Computation: Practice and Experience 2008 apr; 20(5):565–575, 10.1002/cpe.1256.
21. Casati F, Ceri S, Pernici B, Pozzi G. Workflow evolution. Data & Knowledge Engineering 1998 jan;24(3):211–238. , https://doi.org/10.1016/s0169-023x(97)00033-5, 10.1016/s0169-023x(97)00033-5.
22. interoperability | Definition of interoperability in English by Oxford Dictionaries;. https://en.oxforddictionaries.com/definition/interoperability, accessed 22 Sep 2018.
23. Tolk A. What Comes After the Semantic Web - PADS Implications for the Dynamic Web. In: 20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06) IEEE;. , 10.1109/pads.2006.39.
24. portability | Definition of portability in English by Oxford Dictionaries;. https://en.oxforddictionaries.com/definition/portability, accessed 22 Sep 2018.
25. Cohen-Boulakia S, Belhajjame K, Collin O, Chopard J, Froidevaux C, Gaignard A, et al. Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. Future Generation Computer Systems 2017;75:284–298.
26. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux J 2014 Mar;2014(239). https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment.
27. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. PLoS ONE 2017 may; 12(5):e0177459, 10.1371/journal.pone.0177459.
28. Möller S, Prescott SW, Wirzenius L, Reinholdtsen P, Chapman B, Prins P, et al. Robust Cross-Platform Workflows: How Technical and Scientific Communities Collaborate to Develop, Test and Share Best Practices for Data Analysis. Data Science and Engineering 2017 Sep; 2(3):232–244, 10.1007/s41019-017-0050-4.
29. Grüning B, , Dale R, Sjödin A, Chapman BA, Rowe J, et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. Nature Methods 2018 jul; 15(7):475–476, 10.1038/s41592-018-0046-7.
30. Docker Hub;. https://hub.docker.com/, accessed 22 Sep 2018.
31. Conda — Conda documentation;. https://conda.io/docs/, accessed 22 Sep 2018.
32. Zenodo - Research. Shared.;. https://zenodo.org/, accessed 22 Sep 2018.
33. GitHub;. https://github.com/, accessed 22 Sep 2018.
34. Goble CA, Bhagat J, Aleksejevs S, Cruickshank D, Michaelides D, Newman D, et al. myExperiment: a repository and social network for the sharing of bioinformatics workflows. Nucleic Acids Research 2010 may;38(suppl_2):W677–W682. , https://doi.org/10.1093/nar/gkq429, 10.1093/nar/gkq429.
35. figshare - credit for all your research;. https://figshare.com/, accessed 22 Sep 2018.
36. Code as a Research Object;. http://mozillascience.github.io/code-research-object/, accessed 22 Sep 2018.
37. Chirigati F, Rampin R, Shasha D, Freire J. ReproZip. In: Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16 ACM Press; 2016. , 10.1145/2882903.2899401.
38. Gomez-Perez JM, Palma R, Garcia-Silva A. Towards a Human-Machine Scientific Partnership Based on Semantically Rich Research Objects. In: 2017 IEEE 13th International Conference on e-Science (e-Science) IEEE; 2017. p. 266–275, 10.1109/eScience.2017.40.
39. Custovic A, Ainsworth J, Arshad H, Bishop C, Buchan I, Cullinan P, et al. The Study Team for Early Life Asthma Research (STELAR) consortium 'Asthma e-lab': team science bringing data, methods and investigators together. Thorax 2015; 70(8):799–801, 10.1136/thoraxjnl-2015-206781.
40. Moreau L, Freire J, Futrelle J, McGrath RE, Myers J, Paulson P. The Open Provenance Model: An Overview. In: Lecture Notes in Computer Science Springer Berlin Heidelberg; 2008. p. 323–326, 10.1007/978-3-540-89965-5_31.
41. Moreau L, Freire J, Futrelle J, Myers J, Paulson P. Governance of the open provenance model 2009 Jun;https://nms.kcl.ac.uk/luc.moreau/papers/governance.pdf.
42. W3C Provenance Incubator Group Wiki - XG Provenance Wiki;. https://www.w3.org/2005/Incubator/prov/wiki/W3C_Provenance_Incubator_Group_Wiki, accessed 22 Sep 2018.
43. Moreau L, Groth P, Cheney J, Lebo T, Miles S. The rationale of PROV. Web Semantics: Science, Services and Agents on the World Wide Web 2015 dec; 35:235–257, 10.1016/j.websem.2015.04.001.
44. Michaelides DT, Parker R, Charlton C, Browne WJ, Moreau L. Intermediate notation for provenance and workflow reproducibility. In: Mattoso M, Glavic B, editors. Provenance and annotation of data and processes, vol. 9672 of Lecture notes in computer science Cham: Springer International Publishing; 2016. p. 83–94, 10.1007/978-3-319-40593-3_7.
45. Pasquier T, Han X, Goldstein M, Moyer T, Eyers D, Seltzer M, et al. Practical whole-system provenance capture. In: Proceedings of the 2017 Symposium on Cloud Computing - SoCC '17 New York, New York, USA: ACM Press; 2017. p. 405–418, 10.1145/3127479.3129249.
46. Giesler A, Czekala M, Hagemeier B, Grunzke R. Uniprov: A flexible provenance tracking system for UNICORE. In: Di Napoli E, Hermanns MA, Iliev H, Lintermann A, Peyser A, editors. High-Performance Scientific Computing, vol. 10164 of Lecture notes in computer science Cham: Springer International Publishing; 2017. p. 233–242, 10.1007/978-3-319-53862-4_20.
47. Benabdelkader A, VanKampen AA, Olabarriaga SD. PROVman: A PROV-compliant toolkit for provenance management. PeerJ PrePrints 2015; 3(e1102v1), 10.7287/peerj.preprints.1102v1.
48. Freire J, Silva CT. Making Computations and Publications Reproducible with VisTrails. Comput Sci Eng 2012 jul; 14(4):18–25, 10.1109/{MCSE}.2012.76.
49. Soiland-Reyes S, Alper P, Goble C. Tracking workflow execution with TavernaProv 2016 jun;, 10.5281/zenodo.51314.
50. Nekrutenko A, Taylor J. Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. Nature Reviews Genetics 2012 sep; 13(9):667–672, 10.1038/nrg3305.
51. Garijo D, Kinnings S, Xie L, Xie L, Zhang Y, Bourne PE, et al. Quantifying reproducibility in computational biology: the case of the tuberculosis drugome. PLoS ONE 2013 nov; 8(11):e80278, 10.1371/journal.pone.0080278.
52. Garijo D, Gil Y, Corcho O. Abstract, link, publish, exploit: An end to end framework for workflow sharing. Future Generation Computer Systems 2017 oct; 75:271–283, 10.

1016/j.future.2017.01.008.
53. Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten simple rules for reproducible computational research. PLoS Comput Biol 2013 oct; 9(10):e1003285, 10.1371/journal.pcbi.1003285.
54. Mohan A, Lu S, Kotov A. Addressing the Shimming Problem in Big Data Scientific Workflows. In: 2014 IEEE International Conference on Services Computing IEEE; 2014. , https://doi.org/10.1109/scc.2014.53, 10.1109/scc.2014.53.
55. Littauer R, Ram K, Ludäscher B, Michener W, Koskela R. Trends in Use of Scientific Workflows: Insights from a Public Repository and Recommendations for Best Practice. International Journal of Digital Curation 2012 oct; 7(2):92–100, 10.2218/ijdc.v7i2.232.
56. Stodden V, McNutt M, Bailey DH, Deelman E, Gil Y, Hanson B, et al. Enhancing reproducibility for computational methods. Science 2016 dec;354(6317):1240–1241. , http://www.sciencemag.org/cgi/doi/10.1126/science.aah6168, 10.1126/science.aah6168.
57. Stodden V, Miguez S. Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research. Journal of Open Research Software 2014 jul; 2(1), 10.5334/jors.ay.
58. Zhao J, Gomez-Perez JM, Belhajjame K, Klyne G, Garcia-Cuesta E, Garrido A, et al. Why workflows break &#x2014 Understanding and combating decay in Taverna workflows. In: 2012 IEEE 8th International Conference on E-Science IEEE; 2012. , 10.1109/escience.2012.6404482.
59. Gymrek M, Farjoun Y. Recommendations for open data science. GigaScience 2016 may; 5(1), 10.1186/s13742-016-0127-4.
60. Ludäscher B. A brief tour through provenance in scientific workflows and databases. In: Building Trust in Information Springer; 2016.p. 103–126.
61. Chen W, Deelman E. Partitioning and scheduling workflows across multiple sites with storage constraints. In: International Conference on Parallel Processing and Applied Mathematics Springer; 2011. p. 11–20.
62. GATK | BP Doc #11165 | Data pre-processing for variant discovery; 2018. https://software.broadinstitute.org/gatk/best-practices/workflow?id=11165, accessed 22 Sep 2018.
63. Alper P, Belhajjame K, Curcin V, Goble C. LabelFlow Framework for Annotating Workflow Provenance. Informatics 2018 feb; 5(1):11, 10.3390/informatics5010011.
64. Ison J, Kalas M, Jonassen I, Bolser D, Uludag M, McWilliam H, et al. EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics 2013 mar; 29(10):1325–1332, 10.1093/bioinformatics/btt113.
65. Castro LJG, Giraldo OL, Castro AG, Dumontier M. Bioschemas: schema. org for the Life Sciences. In: SWAT4LS; 2017. .
66. Garijo D, Gil Y, Corcho O. Towards Workflow Ecosystems through Semantic and Standard Representations. In: 2014 9th Workshop on Workflows in Support of Large-Scale Science IEEE; 2014. p. 94–104, 10.1109/{WORKS}.2014.13.
67. Alterovitz G, Dean DA, Goble C, Crusoe MR, Soiland-Reyes S, Bell A, et al. Enabling Precision Medicine via standard communication of NGS provenance, analysis, and results. bioRxiv 2018;, 10.1101/191783.
68. Sefton P, Lynch M, Devine G, Loxton D. DataCrate: a method of packaging, distributing, displaying and archiving Research Objects Workshop on Research Objects (RO2018) at IEEE eScience 20182018 Jul;, 10.5281/zenodo.1312323.
69. Data on the Web Best Practices;. https://www.w3.org/TR/dwbp/#ReuseVocabularies, accessed 22 Sep 2018.
70. Kaushik G, Ivkovic S, Simonovic J, Tijanic N, Davis-Dusenbery B, Kural D. Rabix: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions. Pac Symp Biocomput 2017; 22:154–165, 10.1142/9789813207813_0016.
71. Voss K, Auwera GVD, Gentry J, Full-stack genomics pipelining with GATK4 + WDL + Cromwell. F1000Research ; 2017, 10.7490/f1000research.1114634.1.
72. Guimera RV. bcbio-nextgen: Automated, distributed next-gen sequencing pipeline. EMBnet j 2012 feb; 17(B):30, 10.14806/ej.17.B.286.
73. researchobject.org;. http://www.researchobject.org/overview/, accessed 22 Sep 2018.
74. Chard K, D'Arcy M, Heavner B, Foster I, Kesselman C, Madduri R, et al. I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. In: 2016 IEEE International Conference on Big Data (Big Data) IEEE; 2016. p. 319–328, 10.1109/{BigData}.2016.7840618.
75. PROV Model Primer; 2013. https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/, accessed 22 Sep 2018.
76. Moreau L, Missier P, Cheney J, Soiland-Reyes S. PROV-N: The Provenance Notation. W3C; 2013.
77. Huynh TD, Jewell M, Keshavarz AS, Michaelides D, Yang H, Moreau L. The PROV-JSON Serialization. A JSON Representation for the PROV Data Model; 2013.
78. JSON-LD 1.0; 2014. https://www.w3.org/TR/json-ld/, accessed 22 Sep 2018.
79. Services EE. Information Storage and Management: Storing, Managing, and Protecting Digital Information. John Wiley & Sons; 2010.
80. Soiland-Reyes S, Gamble M, Haines R. Research Object Bundle 1.0 2014;, 10.5281/zenodo.12586.
81. Soiland-Reyes S, Khan FZ, Crusoe MR, common-workflow-language/cwlprov: CWLProv 0.6.0. Zenodo ; 2018, 10.5281/zenodo.1471585.
82. The Wfdesc Ontology; 2016. http://wf4ever.github.io/ro/2016-01-28/wfdesc/, accessed 22 Sep 2018.
83. The Wfprov Ontology; 2016. http://wf4ever.github.io/ro/2016-01-28/wfprov/, accessed 22 Sep 2018.
84. Cao Y, Jones C, Cuevas-Vicenttın V, Jones MB, Ludäscher B, McPhillips T, et al. ProvONE: extending PROV to support the DataONE scientific community;.
85. Soiland-Reyes S, Cáceres M. The Archive and Package (arcp) URI scheme. Internet-Draft 2018 jan;.
86. Soiland-Reyes S, Cáceres M. The Archive and Package (arcp) URI scheme 2018 Jul;Preprint, to appear in Proceedings of 2018 IEEE 14th International Conference on e-Science (e-Science), doi = 10.5281/zenodo.1320264, url = http://s11.no/2018/arcp.html.
87. PROV-N: The Provenance Notation; 2013. https://www.w3.org/TR/prov-n/#expression-Agent, accessed 22 Sep 2018.
88. prov 1.5.2; 2018. https://pypi.org/project/prov/, accessed 22 Sep 2018.
89. common-workflow-language/cwltool:Common Workflow Language reference implementation; 2016. https://github.com/common-workflow-language/cwltool#cwl-tool-control-flow, accessed 23 Sep 2018.
90. Robinson M, Soiland-Reyes S, Crusoe MR, Goble C. Common Workflow Language Viewer; 2017. https://view.commonwl.org/.
91. Dobin A, Gingeras TR. Mapping RNA-seq reads with STAR. Current protocols in bioinformatics 2015;51(1):11–14.
92. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, et al. A survey of best practices for RNA-seq data analysis. Genome Biology 2016 jan;17(1). , https://doi.org/10.1186/s13059-016-0881-8, 10.1186/s13059-016-0881-8.

93. TopMed RNA-seq workflow; 2018. https://github.com/FarahZKhan/cwl_workflows/blob/cwlprov_testing/topmed-workflows/TOPMed_RNAseq_pipeline/rnaseq_pipeline_fastq.cwl, accessed 23 Sep 2018.
94. heliumdatacommons; 2017. https://github.com/heliumdatacommons, accessed 23 Sep 2018.
95. Data Commons | NIH Common Fund; 2018. https://commonfund.nih.gov/commons, accessed 23 Sep 2018.
96. Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 2016 mar;3:160018. , http://www.nature.com/articles/sdata201618, 10.1038/sdata.2016.18.
97. heliumdatacommons/cwl_workflows: Example CWL Workflows that run on team Helium PIVOT architecture.;. https://github.com/heliumdatacommons/cwl_workflows, accessed 03 Oct 2018.
98. Trans-Omics for Precision Medicine (TOPMed) Program | National Heart, Lung, and Blood Institute (NHLBI); 2014. https://www.nhlbi.nih.gov/science/trans-omics-precision-medicine-topmed-program, accessed 23 Sep 2018.
99. Gtex RNA-seq pipeline; 2017. https://github.com/broadinstitute/gtex-pipeline/tree/master/rnaseq, accessed 23 Sep 2018.
100. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics 2012 oct;29(1):15–21. , https://doi.org/10.1093/bioinformatics/bts635, 10.1093/bioinformatics/bts635.
101. Tool documentation: MarkDuplicates;. http://broadinstitute.github.io/picard/command-line-overview.html#MarkDuplicates, accessed 23 Sep 2018.
102. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The sequence alignment/map format and SAMtools. Bioinformatics 2009;25(16):2078–2079.
103. DeLuca DS, Levin JZ, Sivachenko A, Fennell T, Nazaire MD, Williams C, et al. RNA-SeQC: RNA-seq metrics for quality control and process optimization. Bioinformatics 2012 apr;28(11):1530–1532, 10.1093/bioinformatics/bts196.
104. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC bioinformatics 2011;12(1):323.
105. Seo JS, Ju YS, Lee WC, Shin JY, Lee JK, Bleazard T, et al. The transcriptional landscape and mutational profile of lung adenocarcinoma. Genome research 2012;.
106. Xu C. A review of somatic single nucleotide variant calling algorithms for next-generation sequencing data. Computational and structural biotechnology journal 2018;.
107. topmed-workflows/topmed-alignment.cwl at cwlprov_testing · FarahZKhan/topmed-workflows;. https://github.com/FarahZKhan/topmed-workflows/blob/cwlprov_testing/aligner/sbg-alignment-cwl/topmed-alignment.cwl, accessed 23 Sep 2018.
108. Data Biosphere; 2018. https://github.com/DataBioSphere, accessed 23 Sep 2018.
109. statgen/docker-alignment: Dockerfile for Alignment; 2017. https://github.com/statgen/docker-alignment, accessed 23 Sep 2018.
110. Abecasis Lab - Genome Analysis Wiki; 2017. https://genome.sph.umich.edu/wiki/Abecasis_Lab, accessed 23 Sep 2018.
111. Cochrane G, Alako B, Amid C, Bower L, Cerdeño-Tárraga A, Cleland I, et al. Facing growth in the European Nucleotide Archive. Nucleic Acids Research 2012 nov;41(D1):D30–D35. , https://doi.org/10.1093/nar/gks1175, 10.1093/nar/gks1175.
112. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint arXiv:13033997 2013;.
113. Introduction to Variant Calling; 2014. https://bioconductor.org/help/course-materials/2014/CSAMA2014/3_Wednesday/lectures/VariantCallingLecture.pdf, accessed 23 Sep 2018.
114. Saunders CT, Wong WSW, Swamy S, Becq J, Murray LJ, Cheetham RK. Strelka: accurate somatic small-variant calling from sequenced tumor–normal sample pairs. Bioinformatics 2012 may;28(14):1811–1817. , https://doi.org/10.1093/bioinformatics/bts271, 10.1093/bioinformatics/bts271.
115. Blue Collar Bioinformatics;. http://bcb.io/, accessed 23 Sep 2018.
116. Somatic Variant Calling Workflow; 2018. https://github.com/FarahZKhan/bcbio_test_cwlprov/blob/master/somatic/somatic-workflow/main-somatic.cwl, accessed 23 Sep 2018.
117. Common Workflow Language (CWL) — bcbio-nextgen 1.1.0 documentation; 2017. https://bcbio-nextgen.readthedocs.io/en/latest/contents/cwl.html#current-status, accessed 23 Sep 2018.
118. Vivian J, Rao AA, Nothaft FA, Ketchum C, Armstrong J, Novak A, et al. Toil enables reproducible, open source, big biomedical data analyses. Nature biotechnology 2017;35(4):314.
119. Nectar Cloud - Nectar;. https://nectar.org.au/research-cloud/, accessed 23 Sep 2018.
120. Kanwal S, Lonie A, Sinnott RO. Digital reproducibility requirements of computational genomic workflows. In: Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on IEEE; 2017. p. 1522–1529.
121. Gruening B, Sallou O, Moreno P, da Veiga Leprevost F, Ménager H, Søndergaard D, et al. Recommendations for the packaging and containerizing of bioinformatics software. F1000Research 2018 jun;7:742. , https://doi.org/10.12688/f1000research.15140.1, 10.12688/f1000research.15140.1.
122. Missier P. The Lifecycle of Provenance Metadata and Its Associated Challenges and Opportunities. In: Building Trust in Information Springer International Publishing; 2016.p. 127–137. , https://doi.org/10.1007/978-3-319-40226-0_8, 10.1007/978-3-319-40226-0_8.
123. Soiland-Reyes S, Khan FZ, common-workflow-language/cwlprov-py: cwlprov-py 0.1.1. Zenodo; 2018. , https://doi.org/10.5281/zenodo.1471376, 10.5281/zenodo.1471376.
124. Carata L, Akoush S, Balakrishnan N, Bytheway T, Sohan R, Selter M, et al. A primer on provenance. Communications of the ACM 2014 may;57(5):52–60. , https://doi.org/10.1145/2596628, 10.1145/2596628.
125. Kim D, Vouk MA. Assessing Run-time Overhead of Securing Kepler. Procedia Computer Science 2016;80:2281–2286.
126. Cristofori A, Bologna I, Gordon J, London SR, Kennedy J, Munich R, et al. Usage Record–Format Recommendation. In: Open Grid Forum; 2013. p. 1–62.
127. Goble C. Better Software, Better Research. IEEE Internet Computing 2014 sep;18(5):4–8. , https://doi.org/10.1109/mic.2014.88, 10.1109/mic.2014.88.
128. Software Carpentry;. https://software-carpentry.org/, accessed 03 Oct 2018.
129. Code Is Science;. http://www.codeisscience.com/, accessed 03 Oct 2018.
130. Khan FZ, Soiland-Reyes S, Crusoe MR, Lonie A, Sinnott R, CWLProv - Interoperable Retrospective Provenance capture and its challenges. Zenodo ; 2018, 10.5281/zenodo.1215611.

131. OBF Travel Fellowship Program | OBF News; 2016. https://news.open-bio.org/2016/03/01/obf-travel-fellowship-program/, accessed 26 Sep 2018.