Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities

Sarah Cohen-Boulakia, Khalid Belhajjame, Olivier Collin, Jérôme Chopard, Christine Froidevaux, Alban Gaignard, Konrad Hinsen, Pierre Larmande, Yvan Le Bras, Frédéric Lemoine, et al.

Scientific Workflows for Computational Reproducibility in the Life Sciences: Status, Challenges and Opportunities

Sarah Cohen-Boulakia[a,b,c], Khalid Belhajjame[d], Olivier Collin[e], Jérôme Chopard[f], Christine Froidevaux[a], Alban Gaignard[g], Konrad Hinsen[h], Pierre Larmande[i,c], Yvan Le Bras[j], Frédéric Lemoine[k], Fabien Mareuil[l,m], Hervé Ménager[l,m], Christophe Pradal[n,b], Christophe Blanchet[o]

[a]*Laboratoire de Recherche en Informatique, Université Paris-Sud, CNRS UMR 8623, Université Paris-Saclay, Orsay, France*
[b]*Inria, VirtualPlants, Montpellier, France*
[c]*Inria, Zenith, Montpellier, France*
[d]*Univ. Paris-Dauphine, PSL Research University, CNRS, UMR 7243, Centre Lamsade, 75016 Paris, France*
[e]*IRISA, Rennes, France*
[f]*INRA, UMR729, MISTEA, F-34060 Montpellier, France*
[g]*Nantes Academic Hospital, CHU de Nantes, France*
[h]*Centre de Biophysique Moléculaire (CNRS UPR4301, Orléans)*
[i]*IRD, DIADE, F-34394 Montpellier, France*
[j]*EnginesOn / INRIA, Rennes, France*
[k]*Institut Pasteur, Unité Bioinformatique Evolutive, Centre de Bioinformatique, Biostatistique et Biologie Intégrative (C3BI, USR 3756 IP CNRS), Paris, France*
[l]*Institut Pasteur, Hub Bioinformatique et Biostatistique, Centre de Bioinformatique, Biostatistique et Biologie Intégrative (C3BI, USR 3756 IP CNRS), Paris, France*
[m]*Institut Pasteur, Centre d'Informatique pour la Biologie, Direction des Systèmes d'Information, Paris, France*
[n]*CIRAD, UMR AGAP, Montpellier, France*
[o]*CNRS, UMS 3601; Institut Français de Bioinformatique, IFB-core, Avenue de la Terrasse, F-91190 Gif-sur-Yvette, France*

Abstract

With the development of new experimental technologies, biologists are faced with an avalanche of data to be computationally analyzed for scientific advancements and discoveries to emerge. Faced with the complexity of analysis pipelines, the large number of computational tools, and the enormous amount of data to manage, there is compelling evidence that many if not most scientific discoveries will not stand the test of time: increasing the reproducibility

of computed results is of paramount importance.

The objective we set out in this paper is to place scientific workflows in the context of reproducibility. To do so, we define several kinds of reproducibility that can be reached when scientific workflows are used to perform experiments. We characterize and define the criteria that need to be catered for by *reproducibility-friendly* scientific workflow systems, and use such criteria to place several representative and widely used workflow systems and companion tools within such a framework. We also discuss the remaining challenges posed by reproducible scientific workflows in the life sciences. Our study was guided by three use cases from the life science domain involving in silico experiments.

Keywords: Reproducibility, Scientific Workflows, Provenance, Packaging environments

---

1. Introduction

Novel technologies in several scientific areas have led to the generation of very large volumes of data at an unprecedented rate. This is particularly true for the life sciences, where, for instance, innovations in Next Generation Sequencing (NGS) have led to a revolution in genome sequencing. Current instruments can sequence 200 human genomes in one week whereas 12 years have been necessary for the first human genome [1]. Many laboratories have thus acquired NGS machines, resulting in an avalanche of data which has to be further analyzed using a series of tools and programs for new scientific knowledge and discoveries to emanate.

The same kind of situation occurs in completely different domains, such as plant phenotyping which aims at understanding the complexity of interactions between plants and environments in order to accelerate the discovery of new genes and traits thus optimize the use of genetic diversity under different environments. Here, thousands of plants are grown in controlled environments, capturing a lot of information and generating huge amounts of raw data to be stored and then analyzed by very complex computational analysis pipelines for scientific advancements and discoveries to emerge.

Faced with the complexity of analysis pipelines designed, the number of computational tools available and the amount of data to manage, there is compelling evidence that the large majority of scientific discoveries will not stand the test of time: increasing reproducibility of results is of paramount

importance.

Over the recent years, many authors have drawn attention to the rise of purely computational experiments which are not reproducible [2, 3, 4, 5]. Major reproducibility issues have been highlighted in a very large number of cases: while [6] has shown that even when very specific tools were used, textual description of the methodology followed was not sufficient to repeat experiments, [7] has focused on top impact factor papers and shown that insufficient data were made available by the authors to make experiments reproducible, despite the data publication policies recently put in place by most publishers.

Scientific communities in different domains have started to act in an attempt to address this problem. Prestigious conferences (such as two major conferences from the database community, namely, VLDB[1] and SIGMOD[2]) and journals such as PNAS[3], Biostatistics [8], Nature [9] and Science [10], to name only a few, encourage or require published results to be accompanied by all the information necessary to reproduce them. However, making their results reproducible remains a very difficult and extremely time-consuming task for most authors.

In the meantime, considerable efforts have been put into the development of *scientific workflow management systems*. They aim at supporting scientists in developing, running, and monitoring chains of data analysis programs. A variety of systems (*e.g.*, [11], [12], [13]) have reached a level of maturity that allows them to be used by scientists for their bioinformatics experiments, including analysis of NGS or plant phenotyping data.

By capturing the exact methodology followed by scientists (in terms of experimental steps associated with tools used) scientific workflows play a major role in the reproducibility of experiments. However, previous work have either introduced individual workflow systems allowing to design reproducible analyses (*e.g.*, [14, 15]) without the aim to draw more general conclusions and discuss the capabilities of the scientific workflow systems to reproduce experiments or it has discussed computational reproducibility challenges in e-science (*e.g.*, [16, 17]) without considering the specific case where scientific workflow systems are used to design an experiment. There is thus a need

[1]International conference on Very Large Data Bases

[2]ACM's Special Interest Group on Management Of Data.

[3]http://www.pnas.org/site/authors/format.xhtml

to better understand the core problematic of reproducibility in the specific context of scientific workflow systems, which is the aim of this paper.

In this paper, we place scientific workflows in the context of computational reproducibility in the life sciences to provide answers to the following key points: How can we define the different levels of reproducibility that can be achieved when a workflow is used to implement an *in silico* experiment? Which are the criteria of scientific workflow systems that make them *reproducibility-friendly*? What is concretely offered by the scientific workflow systems in use in the life science community to deal with reproducibility? Which are the open problems to be tackled in computer science (in algorithmics, systems, knowledge representation etc.) which may have huge impact in the problems of reproducing experiments when using scientific workflow systems?

Accordingly, we make the following five contributions: We present three use cases from the life science domain involving in silico experiments, and elicit concrete reproducibility issues that they raise (Section 2). We define several kinds of reproducibility that can be reached when scientific workflows are used to perform experiments (Section 3). We characterize and define the criteria that need to be catered for by *reproducibility-friendly* scientific workflow systems (Section 4). Using the framework of the criteria identified, we place several representative and widely used workflow systems and companion tools within such a framework (Section 5). We go on to discuss the challenges posed by reproducible scientific workflows in the life sciences and describe the remaining opportunities of research in several areas of computer science which may address them in Section 6 before closing the paper in Section 7.

2. Use Cases

This paper starts with a set of three use cases, extracted from real projects, where scientific workflow systems are used to manage data analyses.

2.1. Next Generation Sequencing for diagnosis in oncology

2.1.1. Context

New and powerful next-generation sequencing (NGS) techniques allow to simultaneously and quickly analyze a large number of genes, up to the entire genome, that are assumed to be involved in diseases. As recently highlighted

[18], the main challenge in applying NGS to medical diagnosis resides in workflow development fulfilling diagnosis interpretation requirements, such as quality control or variant knowledge annotation.

In this context, the preeminent French health and science agency, National Cancer Institute (INCa), is in charge of cancer control in France. The goal of the INCa is to generalize existing workflows designed for diagnosis in oncology, and deploy them in most French hospital laboratories.

2.1.2. Computational tools used

In INCa, workflows are implemented through both very specific chaining tools using command-lines and workflow systems (Galaxy). As such workflows are used in production (and for diagnosis purpose), a particular attention has been paid in deploying solutions allowing different users to (virtually) work in the same run-time computational environment, ensuring in particular that the exact same version of tools and packages is available.

2.1.3. Reproducibility needs

Workflow run-time environment maintenance. A given workflow in INCa usually embeds a large variety of tools and makes use of several databases, evolving quickly. Each version change implies the update, development and validation of a new stable version of the workflow. In addition to the *classical* workflow maintenance, the workflow environment has to be captured again (packaged) for the workflow to be concretely in use again in the laboratories.

Sharing sample data. Reuse of workflows follows very strict rules in particular because given accreditations have to be obtained (following specific norms in the representation and the documentation of the workflow such as CE-IVD or ISO 13485 norms). As part of the requirements to permit reuse, workflows designed in InCA should be accompanied with a gold standard providing a sample data set to be used as input of the workflows and the expected outputs, so that the workflow can be *tested*. Since laboratories process samples from patients, such data sets cannot be used as a standard for quality control due to both genome variability and confidentiality issues. Internal controls can be used in a routine context, but do not constitute a sufficient reference, since they do not cover all cases, and are differently sequenced across laboratories. An external set of samples is thus required, offering a gold standard to ensure workflow reuse across laboratories. This set could be a shared pool of samples provided by laboratories developing workflows.

2.2. Plant Phenotyping

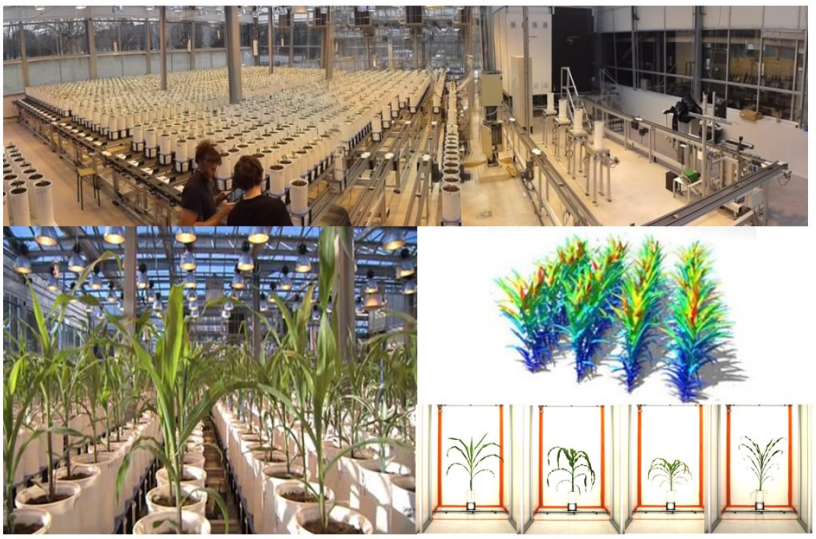

Figure 1: Phenoarch Phenotyping Platform

2.2.1. Context

This use case takes place in the context of the "Phenome"[4] project where high-throughput phenotyping platforms (Figure 1) are used to help assess the genetic variability of plant responses to environmental conditions by allowing growing and automatic extraction of traits of a large number of plants. In such platforms, 1,650 plants are imaged from 12 lateral and one apical view, producing 20,000 images per day and about 3 Terabytes of raw data per experiment. This raw data (images) should be processed to extract traits information (size of the leaves, number of leaves...) and then further analyzed with a response model to simulate the behavior of plants under climate change. Each of these steps combines tools from very different fields of research including image analysis, plant physiology, and crop modeling.

[4]https://www.phenome-fppn.fr/

2.2.2. Computational tools used

All computational experiments performed in PhenoArch are designed using the *OpenAlea* [19] workflow system, widely used in the Plant modeling community. Workflows analyze images and combine raw data with numerous and heterogeneous data sets coming from various external databases or produced by previous workflow runs. Both raw data sets and data sets analyzed by workflows are stored in a dedicated database. As for the tools used in the current workflows, they are parts of two versioned Python packages: `phenomenal`[5] for image analysis and `alinea` [6] for eco-physiological simulations. These packages are platform-independent and can be downloaded and installed to replay an analysis on a local desktop.

2.2.3. Reproducibility needs

Designing and sharing tools. A large variety of algorithms may be used to extract relevant information from images. Dedicated plant phenotype commercial packages use various functions to estimate, for instance, the total plant volume and leaf area. A new landscape of models is emerging in several communities [20, 21, 22]. They make the link between the reconstructed plant architecture and eco-physiological models and allow to perform a more precise analysis of plant traits in varying environmental conditions. These models integrate plant specific knowledge and are often developed or adapted by biologist end-users. It is thus particularly important to initially allow workflow designers to encapsulate such models into (steps of) workflows to be used and shared with biologist end-users.

Ensuring compatibility of tools' versions. Image analysis workflows may be completed by other workflows that couple data analysis with a model for analyzing the response of plant expansion rate to temperature and water availability. In such large workflows, tools are usually designed based on various packages or libraries which may evolve very quickly. Updating analysis workflows becomes tricky when several tools, each based on different versions of the same library, have to be combined.

Tracing the workflow history (citation). Workflows can be copy-pasted and modified to be (partly) reused. In this process, the trace of the original

[5] https://github.com/openalea/InfraPhenoGrid

[6] http://openalea.gforge.inria.fr/wiki/doku.php?id=packages:alinea:alinea

workflows (and the reference to their original designers) is lost. Means to automatically keep track of the sets of workflows reused would increase the willingness to share.

Tracking Data to chain analyses. When workflows are run, they generate large amounts of data which may, in turn, be reused as input of another workflow. For results produced by such a workflow to be interpreted and understood properly (in particular to understand outliers or unexpected results), it is particularly helpful to know how each (reused) data set has originally been produced.

2.3. Transcriptomics analyses

2.3.1. Context

While genomics studies focus on identifying DNA variations, transcriptomics studies focus on gene expression (RNA) in tissues, organs, or, more generally, biological samples. RNA sequencing experiments (RNA-seq) can be seen as an observation of gene expression under multiple biological conditions. A typical RNA-seq experiment aims at linking phenotypes (such as the response to a treatment, at possibly multiple time-points) to genes. RNA-seq experiments can especially be useful in the context of multiple pathologies. A common RNA-seq workflow can thus be used in the context of multi-centric cancer studies. For gene expression results to be interpreted at the scale of a multi-centric study, the same RNA-seq workflow has to be reused by all participating sites. It is also required to share and reuse site-specific datasets to assess the relevance of the data analysis pipeline.

However, designing and sharing RNA-seq analyses is particularly challenging both in terms of computing time, massively produced data and multiplicity and diversity of tools available [23]. In this context, SyMeTRIC is a federated project in Systems Medicine, launched to standardize RNA-seq data analysis pipelines and allow the reuse of RNA-Seq experimental results.

2.3.2. Computational tools used

The SyMeTRIC project relies on two kinds of solutions to represent *in silico* experiments. First, Galaxy is used to design and evaluate a common RNA-seq workflow in the context of multi-centric cancer studies. Its Web-based platform is useful to the SyMeTRIC users who want to learn bioinformatics tools and re-execute them on reference datasets. Second, when large-scale data analysis campaigns with hundreds of biological samples have

to be run, Make-like systems (similar to NextFlow) are being used. While using such approaches requires more technical expertise, they undoubtedly offer versatility and scalability in the tuning of large-scale experiments [24].

2.3.3. Reproducibility needs

Annotation on workflows and runs. Workflows implemented using scripts are generally not annotated with any metadata. As for Galaxy workflows, no standardized metadata is currently in use to annotate workflows either. The same occurs at the execution level: no provenance module is available yet to inspect or query execution traces based on standard schemas or vocabularies.

This situation has two kinds of consequences. First, querying workflows or workflow runs based on keywords/metadata reflecting the nature of data processing tools or the nature/role of processed data is impossible. Second, trusting new experiments (without any comparison point with previous experiment) remains difficult.

Reusing parts of executed workflows. SyMeTRIC users may want to reuse a workflow which has been already run in part, to continue the execution with new processing tools, without re-executing the full workflow (but only the new part). The workflow system used in not necessarily the same, either.

Two points should be noticed. First, while continuing a previously run workflow is possible with scripts supporting check-pointing, it is not always feasible in workflow environments. Second, workflow systems are currently poorly inter-operable. Connecting two data analysis workflows, each based on different systems, generally requires high technical skills.

Alternative workflows to reach the same scientific conclusion. Some technological evolutions may have huge impacts on the concrete implementations of data analysis workflows. In transcriptomics, the same biological question may be answered relying on two technologies, microarray and RNA-seq, which data analysis will concretely involve different computational tools. To foster the reuse of workflows and produced data, it is thus a major requirement to identify relevant workflows addressing the same biological questions, but grounded in different contexts, namely microarray or RNA-seq in this use case.

3. Levels of reproducibilty

The use cases presented in the previous section exhibit different reproducibility needs. These can be better placed by examining the levels of reproducibility and reuse described in the literature. We present in this section such levels of reproducibility. We then introduce definitions of such levels in the specific context of use of scientific workflow systems.

3.1. Discussion on the concepts associated with reproducibility

Reproducibility has obviously been studied in science in larger contexts than computational reproducibility, in particular where wet experiments are involved, and especially in the context of preclinical studies (*e.g.*, [25, 26, 27, 28], see [29] for a review on this topic). A plethora of terms are used including *repeat*, *replicate*, *reproduce*, *redo*, *rerun*, *recompute*, *reuse* and *repurpose etc.* to name a few. We introduce the terminology we will follow in this paper based a review of the various definitions provided in the literature [2, 30, 17, 31, 16, 32].

In such definitions, we need to distinguish the *results* of an *in-silico* experiment, which are concrete datasets produced, from the *conclusion* that can be drawn from the interpretation of such data sets. In particular, the same conclusion can be obtained by two results, both allowing to validate the same (*e.g.*, biological) hypothesis. We distinguish four concepts, which may be considered as several *levels* of reproducibility, as described here-after.

Repeat. A wet experiment is said to be *repeated* when the experiment is performed in the same *lab* as the original experiment, that is, on the same scientific environment [30]. By analogy, an *in silico* experiment is said to be *repeated* when it is performed in the same computational setting as the original experiment. The major goal of the *repeat* task is to check whether the initial experiment was correct and can be performed again. The difficulty lies in recording as much information as possible to repeat the experiment so that the same conclusion can be drawn. As discussed in [17], the granularity at which information (experiments, data sets, parameters, environment) should or could be recorded. The key point is to determine the right balance between the effort to make in recording information and the capability of obtaining very identical results.

Replicate. A wet experiment is said to be *replicated* when the experiment is performed in a different (wet) "lab" than the original experiment. By analogy, a *replicated in silico* experiment is performed in a new setting and computational environment, although similar to the original ones). When replicated, a result has a high level of robustness: the result remains valid in a similar (even though different) setting has been considered. A continuum of situations can be considered between a repeated and replicated experiments.

Reproduce. Reproduce is defined in the broadest possible sense of the term and denotes the situation where an experiment is performed within a different set-up but with the aim to validate the same scientific hypothesis. In other words, what matters is the conclusion obtained and not the methodology considered to reach it. Completely different approaches can be designed, completely different data sets can be used, as long as both experiments converge to the same scientific conclusion. A reproducible result is thus a high-quality result, confirmed while obtained in various ways.

Reuse. A last very important concept related to reproducibility is *Reuse* which denotes the case where a *different* experiment is performed, with similarities with an original experiment. A specific kind of reuse occurs when a single experiment is reused in a new context (and thus adapted to new needs), the experiment is then said to be *repurposed*.

Notice. Repeat and *replicate* may appear to be technical challenges compared to *reproduce* and *reuse* which are obviously the most important scientific targets. However, before investigating alternative ways of obtaining a result (to reach reproducibility) or before reusing a given methodology in a new context (to reach reuse), the original experiment has to be carefully tested (possibly by reviewers and/or any peers), demonstrating its ability to be at least repeated and hopefully replicated [3, 17].

3.2. Levels of reproducibility when workflows are used

We now introduce definitions of reproducibility concepts in the particular context of use of scientific workflow systems.

In such definitions, we clearly distinguish six components of an analysis designed using a scientific workflow. (i) S, the workflow specification, providing the analysis steps associated with tools, chained in a given order, (ii) I, the input of the workflow used for its execution, that is, the concrete data sets and parameter settings specified for any tools, (iii) E, the workflow

context and runtime environment, that is, the computational context of the execution (OS, libs, *etc.*). Additionally, we consider R and C, the result of the analysis (typically the final data sets) and the high level conclusion that can be reached from this analysis, respectively.

Definition (Repeat - using scientific workflows). Given two analyses A and A' performed using scientific workflows, we say that A' *repeats* A if and only if A and A' are identical on all their components.

Definition (Replicate - using scientific workflows). Given two analyses A and A' performed using scientific workflows, we say that A' *replicates* A if and only if they reach the same conclusion while their specification and input components are similar (see [33] for a discussion on workflow similarity) and other components may differ (in particular no condition is set on the run-time environment).

Terms such as *rerun*, *re-compute* typically consider situations where the workflow specification is unchanged.

Definition (Reproduce - using scientific workflows). Given two analyses A and A' performed using scientific workflows, we say that A' *reproduces* A if and only if they reach the same conclusion. No condition is set on any other components of the analysis.

Definition (Reuse - using scientific workflows). Given two analyses A and A' performed using scientific workflows, we say that A' *reuses* A if and only if the specification or input of A' is part of the specification or input of A'. No other condition is set, especially the conclusion to reach may be different.

3.3. Levels of reproducibility in the use cases

We place the needs expressed in the use cases in the context of the reproducibility and reuse levels introduced.

The purpose of the *NGS* use case is patient diagnosis. The workflows are concretely used in production in hospitals. The use case is thus mainly focused on the *repeat* level to ensure that the exact same experiment can be driven. This use case is also related to the *replicate* level to handle different pools of patients.

The *Plants phenotyping* use case describes needs to rerun a given experiment in novel environment or to test the robustness of the approach (over

time, while tools may have changed, while similar tools and data sets are used). It is mainly focused on *replication* and *reproducibility*.

The *Transcriptomics* analyses use cases aims at enhancing collaboration and sharing between different partners, discovering alternative methodologies to augment the confidence one may have on a scientific result. It is mainly focused on *reuse* (repurpose) and *reproducibility* in the broader sense of the term.

4. Reproducible-friendly criteria for scientific workflow management systems

Scientific workflow systems have very different shapes and features, making them not equivalent in the context of reproducibility. In this section we introduce a set of criteria playing a major role in the ability of an *in silico* experiment to be reproducible. Specifically, we tease apart the criteria that need to be catered for when (i) specifying workflows, (ii) executing them, and (iii) packaging them considering the context and runtime environment with the reproducibility levels (requirements), elicited in the previous section, in mind.

4.1. Workflow specification

The workflow specification provides the definition of the data-driven (or scientific) workflow, typically specified as a graph, in which the nodes represents steps, *i.e.*, processing units, that carry data manipulations and transformations, and the edges represent data dependency (and eventually control dependency) links.

Workflow modularity. Workflow specifications can become complex when the target *in silico* experiments involve a large number of data processing and analysis steps. While such workflow specifications can be utilized for verifying repeatability, they are not amenable for checking replicability and reproducibility. To overcome this problem, workflow systems need to provide designers with the means to *modularise* workflow specifications to group similar workflow steps or steps that are strongly coupled into modules, also known as *subworkflows* or *nested workflows*. As well as facilitating the understanding, replicability and reproducibility of workflows, modularity promotes reuse. Designers are more likely to utilize each other's subworkflows, instead of single processing steps, when building new workflows.

Heterogeneity of workflow languages. Workflows shared within a workflow repository are specified using different workflow languages, each dedicated to the workflow system they belong to. This represents a real impediment towards workflow reuse: impossibility to interpret a workflow written in an unknown language, impossibility to repurpose and thus reuse it. This calls for methods for supporting the reuse of heterogeneous workflows, *e.g.*, by developing a standard workflow language that acts as a *lingua franca* between workflow systems, and/or tools for translating workflow specifications to be conform to a targeted workflow language.

Availability of the source/executable code of workflow steps. Some workflow systems adopt a controlled approach whereby the workflow specification embed the specification and/or executable of the software programs that implement the steps of the workflow. This approach ensures the repeatability and replicability of workflows at least within the host workflow system. There are also workflow systems that provide workflow designers with the means to embed calls to software programs that are hosted remotely, and are outside the control of the workflow system (*e.g.*, in the form of web services or services hosted in a cloud environment). While such workflow systems offer a wider range to users when it comes to selecting the software programs to implement the steps of the workflow, they do not guarantee the repeatability of the workflows. Indeed, there is generally no agreement that compels third party providers to continuously supply their software (services), and as such they often decide to stop supplying their services without previous notice (see [34] for more details of the first empirical study dedicated to this problem).

Workflow annotations. Sharing workflow specifications is not sufficient for enabling reproducibility and reuse: other elements such as annotations describing the workflow, its component steps, and the software programs implementing the steps are needed. The annotations can take the form of free text, tags or concepts taken from controlled vocabularies, or a mixture of the two. The availability of annotations helps users understand the experiments implemented by the workflow and assist users in the exploration and search of workflow repositories to identify the workflows relevant to their data analyses.

4.2. Workflow execution

The majority of scientific workflow systems are instrumented to capture information about the execution of a workflow, including the inputs and out-

put of the workflows, the intermediary data artifacts produced by the steps of the workflow. This information allows users to debug workflow specifications, understand the lineage path that leads to a given result, and analyze the workflow in various ways.

Heterogeneity of languages for capturing workflow execution. The heterogeneity of the languages (formats) adopted by workflow systems to encode their executions can be an impediment for assessing the reproducibility of experiments. In particular, comparing the execution traces of equivalent workflows that are ran on different workflow systems become challenging if such systems use different languages for encoding workflow executions. Adopting standards languages for encoding workflow executions is, therefore, desirable to support workflow reproducibility.

Annotation of workflow executions. Workflow execution graph tends to be much larger than workflow specification when the latter contains iterations ('for each' or 'repeat' loops). This calls for mechanisms that helps the user explores workflow execution graphs. Descriptions that annotate the nodes of the workflow execution graph can assist the user in the exploration of such graphs, their understanding, and ultimately reproducibility. Such annotations can be automatically injected by the workflow system or manually added by a workflow user, especially for domain-dependent annotations.

Presentation of workflow executions. To allow users take advantage of recorded workflow executions, there is a need for facilities that present such executions in a user-friendly manner. For example, the user should be able to examine for a given workflow execution, the results produced and identify for each of them the contributing data inputs, among other things. Such capabilities assist users in the task of interpreting and comparing workflow executions, which are required for verifying reproducibility.

4.3. Workflow context and runtime environment

Workflow specifications and associated execution runs are by no means sufficient to guarantee their reproducibility. Indeed, workflows are executed within (and depends on) a runtime environement, *e.g.*, operating system, cloud, and used to test a scientific hypothesis. We therefore need to capture, or more specifically package, information about the runtime environement, and the scientific context to enable reproducibility. Accordingly, we distinguish between two needs to cater for different reproducibility levels:

Need for system-wide packaging. To be able to repeat the experiment as it was carried out by the scientist, we need information about the operating system and/or the cloud environement hosting the workflow system, the libraries/software utilized for running the workflow system, as well as information about the libraries/software needed for running the software/scripts that implement the steps of the workflow. In short, we need to freeze and package the runtime environement of the workflow system.

Need for scientific-wide packaging. Although being able to re-compute an experiment is a major step towards reproducibility, it does not cater for *replicability* and *reproducibility*. To achieve these, we need to capture contextual information that describes the scientific study that involves the workflow(s), the hypothesis that is examined by running the workflows, and the conclusions made by the scientist after examining workflow executions.

5. Workflow system and companion tools faced with reproducibility and reuse: Status

In the first subsection, we review standards, models and tools that were proposed in the last few years to cater for some of the workflow reproducibility needs presented in Section 4. The next subsection is dedicated to the evaluation of workflow systems on such criteria.

5.1. Companion tools and standards

5.1.1. Workflow interoperability

To promote workflow reuse, a number of initiatives has been launched to deal with the heterogeneity of workflow languages (introduced in Section 4.1). In particular, in the context of the SHIWA project [35], Plankensteiner *et al.* proposed a workflow specification language called IWIR [36] to allow portability between major workflow management systems. To annotate (and complement) IWIR workflows, Cerezo and Montagnat [37] proposed a conceptual workflow language for describing the high-level scientific tasks. To promote interoperability, a consortium composed of multiple organizations and workflow system vendors has recently proposed the Common Workflow Language (CWL) [38] [7], with the objective to promote workflow specification portability. CWL is undergoing development.

[7] `https://github.com/common-workflow-language`

5.1.2. Standards and models for workflow executions

Achieving interoperability between workflow systems requires the use of a common language/standard for describing their executions. This is witnessed by the recent efforts made by the suppliers of major scientific workflow systems to adopt the Open Provenance Model and later on the W3C PROV recommendation. PROV does not provide all the concepts necessary for modeling workflow executions. Because of this, we have seen the rise of a handful of PROV-compliant languages for capturing information about workflow executions, *e.g.*, OPMW[8], wfprov[9], prov-wf [39], ProvONE[10]. These languages differ in the kind of information they capture. That said, they tend to agree on the PROV concepts used or extended to model the core concepts and relationships of workflow executions, namely activities and entities, and the usage and generation relationships relating them.

5.1.3. Annotating workflows and their executions

A number of proposals have been made to help users understand workflow specifications and the execution graphs obtained as a result of their execution. For example, Garijo *et al.* [40] proposed an ontology of workflow motifs for annotating workflow specifications, and Missier *et al.* [41] showed how the nodes representing data products in the workflow execution graph can be annotated by leveraging the annotations made at the level of workflow specification.

5.1.4. Workflow repositories

To promote workflow reuse, and ultimately reproducibility, a number of scientific workflow repositories, *e.g.*, CrowdLabs [42], SHIWA [43], the repositories offered with Kepler [44] and Galaxy [45] and myExperiment [46], have been launched to allow scientists to share, publish their workflows for the benefit of the scientific community. As well as workflow repositories, there exists nowadays software catalogs, such Bio.tools [47] and Biocatalogue [48] for sharing information about the software products that can be used to implement the steps of workflows. Both kinds of catalogs are accompagnied with annotations, using free texts and/or controlled vocabularies, that describe the workflows/software products. For example, Bio.tools utilizes con-

[8]http://www.opmw.org/model/OPMW/

[9]`http://purl.org/wf4ever/wfprov`

[10]`http://vcvcomputing.com/provone/provone.html`

cepts from domain ontologies, *e.g.*, the EDAM[11], to annotate the operations of a given software tool and their input and output parameters.

5.1.5. Workflow context and runtime environments

Several approaches have been proposed to preserve workflow runtime environment and their scientific context.

System-wide packaging solutions. Virtualization technologies, *e.g.*, VMware, KVM, VirtualBox and Vagrant can be used to package, "freeze", and expose the workflow working environments. These thechniques are expensive in that they require to copy all the runtime environment. *Containers* technologies such as OpenVZ[12], LXC[13], Docker [49] or Conda[14] represent plausible lightweight alternatives. They only capture the specific sotware dependencies required by applications and share the low-level components provided by the operating system. Containers come with support for their assembly, *e.g.* Docker files, and their composition, versioning and sharing. Other approaches like ReproZip [50] and CDE [51] allow capturing the command line history and associated input and output files, and package them in the case of Reprozip in the form of workflows.

Scientific-wide packaging solutions. Other solutions directly address scientific reproducibility, *e.g.*, Research Objects (RO) [52] or Investigation/Study/Assay (ISA) [53] which capture and expose auxiliary information about the hypothesis investigated and the conclusions made by the scientists. Such approaches create self-contained and query-able packages capturing the required i) data, ii) metadata (including provenance) and iii) protocol/workflow specifications.

5.2. Evaluation of representative workflow systems in the context of reproducibility

We have chosen to present in more details in this section Taverna, Galaxy, OpenAlea, and Nextflow as major representatives of the variety of scientific workflows used to analyze life science data. More precisely, the set of systems has been selected based on three main criteria. First-of-all, all systems are

[11] `http://edamontology.org/`
[12] https://openvz.org/
[13] https://linuxcontainers.org/
[14] `http://conda.pydata.org`

currently in use in large life science projects (and in particular, they are used in our use cases). Second, we do not consider the family of systems focusing on distributed scheduling, which have been studied in detailed in [54] and pose another spectrum of challenges. Third, we restrict the number of systems presented to have one main representative of each kind of workflow system. For instance, systems such as Kepler, Wings, or Knime have common features with one or several workflow systems described in this section.

A summary and comparative information on the systems with respect to reproducible-friendly criteria for workflow specification, execution and environment are respectively provided in the following Tables 1, 2, and 3 (in such tables, a hyphen (-) means that the criteria in question is not supported by the workflow system). In addition to the four workflow systems mentioned above, tables provide information on VisTrails, a workflow system very popular in the neurosciences, with many interesting workflow features related with reproducibility.

	Taverna	Galaxy	OpenAlea	VisTrails	Nextflow
Workflow modularity	nested workflows	nested workflows	nested workflows	nested workflows	- -
Workflow language	XML taverna ontology (SCUFL2)	JSON dedicated file format	Python dedicated file format	XML dedicated file format	Dedicated file format
Interoperability support	extends PROV	A fork of Galaxy uses CWL	export to CWL	-	-
Workflow steps: Local lib tools Command line tools Remote service tools	 Java, R yes yes	 - yes no	 Python yes no	 Python yes yes	 Groovy yes no
Access to source code of workflow steps	local tools only	-	yes	local tools only	local tools only
Workflow annotation	free-text (tags) for describing the workflow and its steps	free-text (tags) for describing the workflow and its steps	-	free-text documentation of the steps	-

Table 1: Workflow specification features

	Taverna	Galaxy	OpenAlea	VisTrails	Netflow
Language	Taverna-PROV	Binary & JSON	provONE	PROV and OPM	Plain text dedicated format
Standard used or extended	PROV	-	PROV	PROV and OPM	-
Annotation	-	-	-	-	-
Presentation	GUI to display and explore saved/latest runs	GUI to explore, share and reuse histories	GUI to explore a given run. Jupyter notebook for linear workflows	GUI, visualize tree of versions and compare runs	Reports traces, performances, and execution DAG

Table 2: Workflow execution features
How the framework records execution provenance and which tools are provided to explore the provenance.

	Taverna	Galaxy	OpenAlea	VisTrails	Nextflow
System-wide packaging	Plans for Docker	Docker, Conda	Conda, Vagrant, plans for Docker	Conda, Vagrant, Docker, ReproZip	Docker
Scientific-wide packaging	Research Objects	Research Objects and ISA	-	-	-

Table 3: Workflow Environment features
Framework used to capture the environment of execution of a workflow.

5.2.1. Taverna

Taverna [55] is an open source scientific workflow management system that provides a workbench for designing workflows, an embedded workflow engine for local execution, and a separate workflow server for remote execution.

Workflow specification. A Taverna workflow is defined in terms of the SCUFL2 language [56], an XML format that extends W3C PROV. Taverna workflows are mainly data driven although for-each loops can be expressed. Processors can be associated with local programs (bean script, R-script) or remote

programs (REST service, cloud service). A user can specify a workflow from scratch or use existing workflows as embedded subworkflows. Taverna is in principle domain-independent. However, it has been used extensively in the life sciences. Taverna also embeds search capabilities to retrieve workflows and tools from myExperiment and biocatalogue, respectively.

Workflow execution. Taverna is instrumented to capture the retrospective provenance of workflow executions using Taverna-PROV, a model that extends PROV and WfPROV[15] to capture features that are specific to Taverna workflows, in particular iterations. Taverna provides a basic result panel for browsing the results of workflow executions. The panel is primarily used to show for a given processor input (respectively output) the data artifacts that were used (respectively generated) during the execution. Sophisticated queries for tracing the lineage of data artifacts or comparing the results of multiple execution of the same or different workflows are not supported.

Context and runtime environment. Taverna does not yet support virtualization using a mechanism such as Docker, although there are current development efforts that will allow users to create a Docker Image that executes a Taverna processor. Taverna also comes with a plugin for using UNICORE grid resources.
Regarding scientfic wide packaging, it has been demonstrated how taverna workflows can be pacakged into reproducible research objects [52].

Taverna: limitations in the context of reproducibility. Taverna allows workflow designers to make use of third party web services in the composition of their workflows. This reliance on *volatile* third party web services means that workflows cannot run because of the unavailability of one or more services used within the workflow in question.

Taverna workflow tends to contain an important number of *shims*, which are processors that are utilized for performing basic data format transformation. Indeed, the reliance on third party web services that adopt different data structures for their input and output generates the need for using shims to resolve the mismatches between services in a workflow. As a result, the workflow specification becomes more complex than it should be, as it contains processors that do not contribute to the overall data analysis implemented

[15] `http://purl.org/wf4ever/wfdesc#`

by the workflow, and therefore may make it more difficult for a potential user to understand the experiment implemented by the workflow based solely on the workflow specification.

Regarding workflow execution presentation, Taverna provides little support for their exploration, e.g., it is difficult to trace the lineage of a given result. This is particularly the case when some of the workflow processors are executed multiple times, through iteration, within a single execution.

Taverna has also limited support for virtualization. Although there are currently efforts by the Taverna team in this direction, Taverna does not provide at the time of writing support for, *e.g.*, deploying Taverna workflows on Cloud environments, to our knowledge.

5.2.2. Galaxy

Galaxy [57, 58] is very popular workflow system in the bioinformatics community.

Workflow specification. Galaxy workflows are internally specified as *composite tasks*, each of which is defined by a tool descriptor specifying the associated software and input and output parameters. Galaxy uses a dedicated JSON format for specifying workflow, although a fork of it utilize the Common Workflow Language (CWL). Galaxy supports nesting of subworkflows. Galaxy uses *Toolshed*, a repository [45] to manage tool versions as well as their dependencies for automated local deployment. It also provides *Galaxy Pages* to document and share the above-mentioned elements in a human-readable form.

Workflow execution. Galaxy uses *Histories*, which trackes workflow executions, through tool configuration parameters, input parameters, including reference datasets [45], and produced results. It also provides a GUI for exploring and sharing workflow executions.

Context and runtime environment. Galaxy allows the use of a number of external solutions to resolve tool dependencies, such as Docker, to provide a persistent and portable execution environment. Regarding scientific-wide packaging, Gonzalez-Beltran *et al.* showed how galaxy workflows can be packaged into reproducible research objects and ISA-TAB experiments [59].

Galaxy: limitations in the context of reproducibility. Galaxy currently encounters four main limitations.

The first limitation is due to the technical complexity of the system relying on specific software and reference datasets that must be locally available with the appropriate version to allow workflow repeat. Although best practices for managing the system can prevent such issues, the use of Toolsheds can lead to duplicated maintenance effort, when the same tools need to be available outside of the Galaxy environment.

A second associated point is that experiments run in Galaxy are difficult to re-run outside Galaxy. This is mainly due to the use of an in-house language for tool and workflow specifications and executions. Such a lack of standardization leads to porting issues, even between two different versions of Galaxy.

Third, there is currently no possibility to reuse such code as building blocks for other analysis, through their inclusion in workflows for instance.

Finally, Galaxy presents some limitation regarding the reuse and sharing of workflows runs and produced results in particular. Galaxy Histories record provenance information, but they do not rely on a reference schema/vocabulary. This limitation seriously hinders the possibilities to query one or multiple histories based on standard vocabularies. This also prevents from exporting and combining them as interoperable provenance traces.

5.2.3. OpenAlea

OpenAlea is an open source scientific workflow system with a visual programming environment that enables users to design, execute and interact with workflows. Being domain-independent, OpenAlea has been used in a variety of domains but its main applications come from the plant science community (*e.g.*, with the simulation of crop physiology and development).

Workflow specification. The architecture of OpenAlea is based on the Python language. Every component in OpenAlea can be accessed through the language, allowing the user to interact with the outputs of a workflow execution, to drive the execution and to extend the system by adding new components dynamically. OpenAlea provides users with the possibility of exporting workflow specifications into a CWL format. OpenAlea tools (*e.g.*, models, workflows, components) are published and shared as Python packages on the web either on public repositories (*e.g.*, PyPi), or through the main OpenAlea

web repository and through dedicated web sites of groups which use and contribute to OpenAlea[16].

Workflow execution. OpenAlea is equipped with a provenance module to keep track of the data items produced during execution. OpenAlea has recently launched a new feature where workflow executions can be exported as notebooks: Each cell of the notebook contains the script of the workflow node executed [60]. The input/output data are direct references to the data produced and stored. When workflow executions have a complex structure (*e.g.*, because loops are involved), this approach is not a suitable solution.

Context and runtime environment. OpenAlea uses the distribution/installation mechanism of Python packages to share both tools and workflows. Workflows and tools are both viewed, in OpenAlea, as components. OpenAlea packages are extensions of classical Python packages. They contain data, tools libraries and workflow specification.

OpenAlea packages can be automatically converted to Debian or rpm packages, because they contain enough meta-information and inherit of the features of Python packages . OpenAlea is thus compatible with several systems, from local virtual environment (*e.g.*, VirtualEnv, Conda) to virtual machines (Vagrant) and Docker containers.

OpenAlea: limitations in the context of reproducibility. Reproducibility in OpenAlea encounters several limitations.

The first limitation is the possible large and hidden complexity of the system which depends on a large and heterogeneous set of interconnected libraries, both provided by the OpenAlea community, but also developed by single users and dynamically discovered by the OpenAlea package manager. While this feature allows end-users to add new packages and share them with the OpenAlea community, several workflows may not be usable if their dependencies are not released as public OpenAlea packages.

A second limitation which concerns both reproducibility and reuse is the lack of central repository for OpenAlea workflows, packages and provenance. Searching for a given tool or workflow is limited to packages already installed on a given computer. This solution guarantees that any workflow found on

[16]See for example the following web sites: http://www.stse-software.org and https://www.cpib.ac.uk/research/themes/digital-plant

a computer will be actually runnable. In return, it hampers the simple exchange of workflows and the discovery of workflows currently in development elsewhere.

A third limitation concerns is with respect to workflow execution presentation. While OpenAlea is able to capture the provenance of execution and expose it to users as electronic notebooks, there is no simple solution to visualize and navigate a huge amount of provenance data in a concise way.

5.2.4. Nextflow

Nextflow[61] is a command-line based workflow system implemented in Groovy [62] (a scripting language for the Java Virtual Machine), developed to simplify design of complex parallel scientific workflows. Unlike workflow systems presented above, it comes with no GUI. Workflows are written in text files, and run in a terminal, or with the help of NextflowWorkbench[63], an integrated development environment (IDE).

Workflow specification. Nextflow is based on the dataflow programming model. A workflow is made of two kinds of nodes: processors (as in Taverna) and operators (that can be seen as *native* processors), and its edges are channels that represent the data flow and connect two nodes.

Nextflow processors are responsible for the execution of scientific computations, and are specified by defining their input and output channels and the computation to execute. Computations are defined as generic scripts (shell, groovy, or any scripting language). Descriptions of processors also specify how the computation will be executed (local mode, HPC platform or cloud). Nextflow operators are used to apply specific transformations to data (*e.g.*, connecting or forking channels, filtering data, *etc.*). Workflows are specified using a dedicated language that, unfortunately, is not based on a standard.

Workflow execution. Workflow execution is deduced from the input and output channels of each processor, and its initial data. If a processor is added or modified, or input data is modified, only computations that are affected by the change are rerun if resume mode is enabled. Moreover, using the cache directive, a processor can specify if it must be rerun each time or not. Nextflow does not use or extend a standard for workflow execution. It does not provides either a means for annotating them. Nextflow does not provide a proper module to query and/or visualize workflow executions.

Context and runtime environment. Nextflow does not directly support tool dependency management. If a tool needed by the workflow is not present in the running system, runs will not complete. However, it uses several systems to minimize the risk of having missing tools, and to keep track of each version of tools that have been run. For example, it is compatible with Docker. Each processor can thus specify in which docker container it must run, or which tool to load from modules. Docker will take care of automatically downloading the containers required for the workflow execution.

Nextflow users are encouraged to use *git* hosting services (*e.g.*, GitLab, BitBucket and GitHub) as workflow repositories. Nextflow can automatically retrieve and execute a workflow located in a public repository on GitHub for example.

Nextflow: limitations in the context of reproducibility. Nexflow encounters three main limitations.

First, as there is no central repository of tools, it is the responsibility of the developer to implement the workflow in a reproducible way. If the workflow does not control the tools used to execute it using Docker module and git, it may be difficult or impossible to reproduce. Second, once the workflow executed, provenance information consists of tags associated with each processor in output log files. As it does not rely on a reference vocabulary, it is not possible to query this data so far. Third, on the one hand, Nextflow is versatile in the sense that it is very easy to install and workflows can be developed very quickly. On the other hand, it does not define any structured provenance information and there is not any GUI interface for building workflows or query their execution logs which make workflow less easy to share and reuse.

Note also that all reviewed systems suffers from the fact that they do not provide the means for annotating workflow execution runs before eventually sharing and publishing them.

6. Challenges and Opportunities

While referencing to the specific problems encountered in the use cases, this section discusses major remaining open challenges related to reproducibility and reuse of experiments implemented using scientific workflows. We clearly distinguish problems associated with social issues from computer

science issues, and focus on the latter. The first subsection is dedicated to problems where partial solutions are available (we clearly underline which are the remaining challenges) while the next subsection highlights even more open issues.

6.1. Partial available solutions

6.1.1. On the representation of Workflow Specification and Execution (Repeat and Replicate)

The previous section has provided information on the state-of-the-art techniques especially addressing points such as how to track workflow specifications and workflow executions (expressed in the plant use cases and, to some extent, also in the transcriptomic use case) describing usable standards both to describe prospective (workflow specification) and retrospective (workflow execution) provenance (families of PROV approaches). There thus exist standards, in particular for provenance information, which are able to represent the description of each workflow and all the elements involved in a given workflow execution.

However, at least three major remaining open problems (directly associated with complex computer science research problems) have to be underlined.

Need for tools to visualize, mine, query huge amounts of provenance information. While a few workflow systems currently use such standards, many systems are still based on in-house workflow and provenance representation as depicted in Tables 1 and 2 mainly because PROV standards are very generic, involve a large variety of terms in standardized vocabularies and to be better understood and used, provenance information samples need to be explored, visualized, queried, compared, and mined. However, except for specific cases, there is currently no interactive systems and tools able to visualize, query or mine provenance information represented using these standards. Faced with the potential huge size of the provenance graphs generated, designing such tools is not as easy as one may think. It directly means developing carefully optimized algorithms to tackle the inherent complexity of graph-related problems (*e.g.*, the (sub)graph isomorphism problem).

Need for interoperability between workflows. Being able to rerun a possibly complex experiment which has been partly run in the past is a need illustrated in the transcriptomics use case. Command-line workflow systems (like

Snakemake [64] or NextFlow) may provide solutions and some workflow systems (like VisTrails or OpenAlea) allow that only part of a given workflow is rerun. However, the major difficulty to enable partial recomputation lies in composing several sub-part of workflows, possibly designed in different systems. While CWL may provide hints (it is still a work under development), workflow interoperability is particularly difficult to achieve (as witnessed by several attempts in this direction, like Tavaxy [65]). The key research point is indeed not to provide a common representation of the workflows (which is already complex faced with the diversity of systems) but to provide a *runnable* representation. This involves carefully defining links between the workflow specification and the workflow execution, possibly designing new models of computation and thus involve research in language theory, compilation, and software engineering. Part of the challenges to be tackled here are thus naturally common to the challenges mentioned in the *Workflow Specification and Execution* paragraph.

Need for reducing workflow execution graphs. Workflow execution graphs tends to be much larger than workflow specifications. To address this issue, several researchers investigated the problem of summarizing the workflow execution graph by exploiting information in the workflow specification. For example, the Zoom system [66] allows the user to identify the elements in the workflow specifications that are of interest. A smaller graph representing the workflow specification is then induced based on the selection made by the user, and utilized to reduce the size of the workflow execution graph. Alper et al. [67] developed a similar approach that exploits semantic annotations describing the steps of the workflow specification. They apply reduction rules to summarize the workflow specification by abstracting away steps that are not relevant from the scientific point-of-view. Other approaches have been proposed in the same spirit, such as [68]. While these three approaches have all the same goal, they use very different techniques, from graph algorithms to web semantics, demonstrating the wide spectrum of research domains that can be used to tackle the problem of *workflow reduction*.

6.1.2. On sharing workflows and tools (Reuse)

As for how tools and workflows can currently be designed and shared and what kind of annotation (from which ontologies) can be provided on workflow specification and provenance (as in the transcriptomic use case), elements of solution are available. The collection of Research Objects Ontologies and the

EDAM ontology, designed in very large collaborative projects, are elements of solution to annotate large amount of tools, workflows and their executions. Such ontologies are currently used in large sharing platforms such as bio.tool and myExperiment. However, for enhancing reuse based on these elements of solutions, several open challenges need to be overcome

Need for approaches to (semi-)automatically annotate tools and workflows with ontology terms. While several ontologies have been designed, there is currently no powerful approach able to assist users in annotating their workflows. Text-mining approaches, in the spirit of [69] would be very helpful to help the annotation task. The challenge lies in designing text-mining and data-mining tools able to make the most of the various facets of workflow's metadata, including the tools used in the workflow, the text-based descriptions of the workflow but also the profile of the workflow uploader (possibly including his/her social network on the sharing platforms, the other workflows s/he may have used or liked in the past).

Need for workflow citation process. Another key need, expressed in the plant use case, is that workflows should be *citeable*, to be referenced when they are reused. Similarly to papers, data sets, or software, several initiatives have actually reached the same conclusion: Workflows, which encapsulate the computational methodology followed to conduct an experiment, should be *citeable objects*. While connected to the problem of history record (a key feature of the VisTrails system), the problem of workflow citation is different especially because the workflows to be considered may be i) very complex graphs, ii) composed of nested workflows and (iii) the result of a combination of several reused workflows (the workflow history is not seen in isolation). While VisTrails focuses on tree structures, one of the key challenge in workflow citation is to consider (possibly arbitrarily complex) acyclic graphs.

6.2. Open Challenges

Several solutions are now available to allow repeating an *in silico* experiment. However, enhancing replication, reproducibility and reuse depends on progress made in several fields of computer science, including databases, knowledge representation but also graph algorithmics, software engineering and systems. While the previous subsection have presented contexts where

elements of solutions were available and have started to highlight both technical and fundamental challenges, the remainder of this section focused on even more open problems.

6.2.1. On Workflow Replication

Workflow maintenance. The NGS use case is related to the major problem of reruning a given workflow in an runtime environment which may have changed. As introduced in 5.1, virtualization techniques are particularly well-suited to repeat an experiment. However, due to the changing nature of the computational experiments, workflows with their dependencies need to be updated to take into account newer versions to benefit of improvements, but also bug and security fixes. Nevertheless, updating a virtual machine is as complex as updating any machine, compatibility of versions between the different tools have to be manually checked. No support is currently provided to automatically select the last working set of dependencies while this is a key point to go from repeat to replicate and reproduce levels.

This point is actually related to the fact that in various workflow systems (such as Galaxy, VisTrails or OpenAlea), providers of tools have no way to know which workflows use their tools. Obviously, good practices from the software community have their role to play here such as pack versions using *Git*, make systematic unit and functional tests on packages, and follow a continuous integration approach to guaranty the quality of packages. However, there is a need for more general algorithms able to determine as efficiently as possible how to update an environment given a set of incompatibilities between packages. Solving this kind of issues may involve research both in algorithmic (to reduce the very large set of *compatible* environments to consider) and software engineering and language (*e.g.*, software product line approaches, *etc.*).

6.2.2. On workflow reuse and reproducibility

Allowing users to test and verify workflows. As for reuse, the transcriptomic and NGS use cases have introduced requirements which may be part of the good practices to be followed for workflow to be reused. Interestingly, providing sample data (which is the focus of the NGS use case) appears to be part of several (high level) good practices and guidelines when designing workflows: in the workflow4ever project [70] the workflow designer is strongly encouraged to "Provide example inputs and outputs" (rule # 4) and "Test and validate" (rule # 9). Very large sharing repositories like BioSharing, as well

as recent FAIR principles [71], have the same kind of expectations (each tool, workflow, standard should be provided with sample data and gold standards).

One requirement is to equip sharing platforms with simple user interfaces allowing users to deposit data sets. However, two important points should be emphasized.

First, one of the most important and open emerging challenges consist in providing techniques to automatically generate small *representative* data sets (to be provided in the platform) from the possibly huge data sets on which the experiment has been initially run. Carefully extracting representative data samples and possibly dealing with privacy issue (that is, considering only a subpart of the data set) may relies on powerful and fine statistical and data mining techniques.

Second, on a more general point-of-view, approaches and tools for systematically allowing workflows to be tested and validated are obviously missing. One of the key point to consider is to better formalize the relationships between workflow systems and scripts. Classical software engineering practices, that are commonly used in software development and popularized recently with initiatives like *Software carpentry* [72], should be used systematically to ensure repeatability of workflow execution. Systematic Unit and functional test on workflow repository as well as continuous integration would provide a measure of the repeatability. Such concepts have to be properly defined in the context of scientific workflows. In the same spirit, methodology of software quality assurance in the context of the entire life cycle of scientific workflow development process need to be envisioned.

Guiding workflow design. Designing user-friendly workflows, which are easy-to-use and understand is a need appearing in all use cases. We consider this problem following two directions, both directly related to the problem of workflow design.

First-of-all, in the previous section, approaches such as [66, 67, 68] have been mentioned as possibilities to reduce the complexity of workflows and make them easier to understand, allowing to hide over technical parts of workflows. Other approaches either reduce the (structural) complexity of workflows, by detecting fragments to be modified such as DistillFlow [73](dedicated to Taverna) or have mined workflow fragments to be reused [74]. More generally speaking, there is a crucial need to imagine approaches and provide concrete tools able to guide workflow design, with the aim of providing

reproducible-friendly workflows that are reusable. While very interesting hints have been discussed in the past [75, 76], no concrete solution is currently in use. Challenges are again numerous and intrinsically involve developing data mining, graph algorithmics and software engineering research approaches to come up with *workflow design patterns*.

Second, workflow design should also be considered in relationship with programming languages, or electronic notebooks (*e.g.* IPython [77], Rstudio [78] and more recently Jupyter [79]), which are largely used in the bioinformatics community to perform *in silico* experiments. Bridging the gap between the use of scripts and workflows is of paramount importance and would have huge impact on reuse. Projects such as noWorkflow [80] and YesWorkflow [81] have started digging in this direction and have provided interesting elements of solutions (as the notion of the provenance of a script).

Workflow Similarity. Last but not least, the ability to discover alternative workflows able to reach the same biological conclusion as the original workflow (expressed in the NGS use case) is of paramount importance. Reproducibility, in the broader sense of the term, is directly expressed here. The central aspect of this question is the definition of similarity in scientific workflows, to enable determining automatically that several workflows have the same kind of functionality. This topic of workflow similarity has been extensively studied in the last years and the concept of workflow similarity is now well understood [82, 83, 84, 85]. However, while a few new efficient algorithms have been designed to search for workflows by similarity in large repositories [86] or to cluster similar workflows, none of them have been deployed in workflow sharing platforms.

7. Conclusion

Reproducibility of *in silico* experiments analyzing life science data is recognized to be a major need. As they provide a means to design and run scientific experiments, scientific workflow systems have a crucial role to play to enhance reproducibility. In this context, the contributions of this paper are five-folds. First, we introduce a set of three use cases, highlighting reproducibility needs in real contexts. Second, we provide a terminology to describe reproducibility levels when scientific workflows are used. Third, we introduce a set of criteria to define reproducibility-friendly workflow systems.

Fourth, we have carefully reviewed the literature and evaluated workflow systems and companion tools based on such criteria. Last, we have highlighted challenges of research in this domain.

Ensuring reproducibility and reuse has obviously to face with numerous social issues (willingness to share, following good practices...). However, and most importantly, we have identified key issues, proper to reproducibility and reuse with scientific workflows, which highly depend on progress made in several research fields of computer science. Scientific workflows and their executions have very complex graph structures. Comparing workflows or executions, storing the history of workflows (*e.g.*, to cite the workflows reused), reducing the structure of such graphs to make them more reusable, are directly related to the design of efficient graph algorithms which are key challenges in graph algorithmics and databases. As for chaining the use of workflows (possibly coming from different systems), rewriting a workflow into another equivalent (but more simple) workflow, optimizing a workflow execution while ensuring its repeatability relies on key issues of languages theory and software engineering. Concerning maintainability of pieces of software in the workflow environment, it relies on progress made on software engineering but also possibly in combinatorial algorithmics since the number of combinations of possible new environments may be huge. Last but not least, designing systems to annotate such complex workflows and their executions and then be able to efficiently retrieve, query and compare them based on their annotations relies on progress made in the semantics web, text mining, and database communities.

8. Acknowledgement

The authors acknowledge the support of GDR CNRS MaDICS, programme CPER Région Bretagne "CeSGO", and programme Région Pays de la Loire "Connect Talent" (SyMeTRIC). We acknowledge funding by the call "Infrastructures in Biology and Health" in the framework of the French "Investments for the Future" (ANR-11-INBS-0012 and ANR-11-INBS-0013). This work was conducted in part at the IBC (Institute of Computational Biology) in Montpellier, France.

[1] E. R. Mardis, A decade's perspective on dna sequencing technology, Nature 470 (2011) 198–203.

[2] V. Stodden, P. Guo, Z. Ma, Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals, PloS one 8 (2013) e67111.

[3] V. Stodden, F. Leisch, R. D. Peng, Implementing reproducible research, CRC Press, 2014.

[4] D. Garijo, S. Kinnings, L. Xie, L. Xie, Y. Zhang, P. E. Bourne, Y. Gil, Quantifying reproducibility in computational biology: the case of the tuberculosis drugome, PloS one 8 (2013) e80278.

[5] L. P. Freedman, I. M. Cockburn, T. S. Simcoe, The economics of reproducibility in preclinical research, PLoS Biol 13 (2015) e1002165.

[6] A. Nekrutenko, J. Taylor, Next-generation sequencing data interpretation: enhancing reproducibility and accessibility, Nature Reviews Genetics 13 (2012) 667–672.

[7] A. A. Alsheikh-Ali, W. Qureshi, M. H. Al-Mallah, J. P. Ioannidis, Public availability of published research data in high-impact journals, PloS one 6 (2011) e24357.

[8] R. D. Peng, Reproducible research and biostatistics, Biostatistics 10 (2009) 405–408.

[9] G. Santori, Journals should drive data reproducibility, Nature 535 (2016) 355–355.

[10] M. B. Yaffe, Reproducibility in science, Science Signaling 8 (2015) eg5–eg5.

[11] J. Goecks, A. Nekrutenko, J. Taylor, Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences., Genome Biology 11 (2010) R86.

[12] T. Oinn, M. Greenwood, M. J. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, et al., Taverna: lessons in creating a workflow environment for the life sciences, Journal of Concurrency and Computation: Practice and experience (2002).

[13] C. Pradal, S. Dufour-Kowalski, F. Boudon, C. Fournier, C. Godin, Openalea: a visual programming and component-based software platform for plant modelling, Functional plant biology 35 (2008) 751–760.

[14] J. Freire, C. Silva, S. Callahan, E. Santos, C. Scheidegger, H. Vo, Managing rapidly-evolving scientific workflows., Proc. of IPAW (2006) 10–18.

[15] C. L. Zheng, V. Ratnakar, Y. Gil, S. K. McWeeney, Use of semantic workflows to enhance transparency and reproducibility in clinical omics, Genome Medicine 7 (2015).

[16] J. Freire, N. Fuhr, A. Rauber, Reproducibility of data-oriented experiments in e-science, in: Dagstuhl Seminar 16041.

[17] J. Freire, P. Bonnet, D. Shasha, Computational reproducibility: state-of-the-art, challenges, and database research opportunities, in: Proceedings of the 2012 ACM SIGMOD international conference on management of data, ACM, pp. 593–596.

[18] M. Meyerson, S. Gabriel, G. Getz, Advances in understanding cancer genomes through second-generation sequencing, Nature Reviews Genetics 11 (2010) 685–696.

[19] C. Pradal, C. Fournier, P. Valduriez, S. Cohen-Boulakia, Openalea: scientific workflows combining data analysis and simulation, in: Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM), ACM, p. 11.

[20] R. T. Furbank, M. Tester, Phenomics–technologies to relieve the phenotyping bottleneck, Trends in plant science 16 (2011) 635–644.

[21] T. B. Brown, R. Cheng, X. R. Sirault, T. Rungrat, K. D. Murray, M. Trtilek, R. T. Furbank, M. Badger, B. J. Pogson, J. O. Borevitz, Traitcapture: genomic and environment modelling of plant phenomic data, Current opinion in plant biology 18 (2014) 73–79.

[22] U. Schurr, F. Tardieu, D. Inzé, X. Dreyer, J. Durner, T. Altmann, J. Doonan, M. Bennett, EMPHASIS – European Multi-environment Plant pHenotyping And Simulation InfraStructure, in: EPPN Plant Phenotyping Symposium, Barcelona, Spain.

[23] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, M. Wojciech Szcześniak, D. Gaffney, L. L. Elo, X. Zhang, et al., A survey of best practices for rna-seq data analysis, Genome Biology (2016).

[24] J. Leipzig, A review of bioinformatic pipeline frameworks, Briefings in Bioinformatics (2016) bbw020.

[25] T. M. Errington, E. Iorns, W. Gunn, F. E. Tan, J. Lomax, B. A. Nosek, An open investigation of the reproducibility of cancer biology research, Elife 3 (2014) e04333.

[26] S. H. Richter, J. P. Garner, C. Auer, J. Kunert, H. Würbel, Systematic variation improves reproducibility of animal experiments, Nature Methods 7 (2010) 167–168.

[27] M. A. Smith, P. Houghton, A proposal regarding reporting of in vitro testing results, Clinical Cancer Research 19 (2013) 2828–2833.

[28] C. G. Begley, L. M. Ellis, Drug development: Raise standards for preclinical cancer research, Nature 483 (2012) 531–533.

[29] C. G. Begley, J. P. Ioannidis, Reproducibility in science improving the standard for basic and preclinical research, Circulation research 116 (2015) 116–126.

[30] C. Drummond, Replicability is not reproducibility: nor is it good science (2009).

[31] C. Goble, Results may vary: reproducibility, open science and all that jazz, in: keynote at ISMB/ECCB, 2013.

[32] S. N. Goodman, D. Fanelli, J. P. Ioannidis, What does research reproducibility mean?, Science translational medicine 8 (2016) 341ps12–341ps12.

[33] J. Starlinger, B. Brancotte, S. C. Boulakia, U. Leser, Similarity search for scientific workflows, PVLDB 7 (2014) 1143–1154.

[34] J. Zhao, J. M. Gómez-Pérez, K. Belhajjame, G. Klyne, E. García-Cuesta, A. Garrido, K. M. Hettne, M. Roos, D. D. Roure, C. A. Goble, Why workflows break - understanding and combating decay in taverna workflows, in: 8th IEEE International Conference on E-Science, e-Science 2012, Chicago, IL, USA, October 8-12, 2012, IEEE, 2012, pp. 1–9.

[35] K. Plankensteiner, R. Prodan, M. Janetschek, T. Fahringer, J. Montagnat, D. Rogers, I. Harvey, I. Taylor, Á. Balaskó, P. Kacsuk, Fine-Grain Interoperability of Scientific Workflows in Distributed Computing Infrastructures, Journal of Grid Computing 11 (2013) 429–456.

[36] K. Plankensteiner, J. Montagnat, R. Prodan, Iwir: A language enabling portability across grid workflow systems, in: Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science, WORKS '11, ACM, New York, NY, USA, 2011, pp. 97–106.

[37] N. Cerezo, J. Montagnat, Scientific workflow reuse through conceptual workflows on the virtual imaging platform, in: Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science, WORKS '11, ACM, New York, NY, USA, 2011, pp. 1–10.

[38] P. Amstutz, R. Andeer, B. Chapman, J. Chilton, M. R. Crusoe, R. V. Guimerà, G. C. Hernandez, S. Ivkovic, A. Kartashov, J. Kern, D. Leehr, H. Ménager, M. Mikheev, T. Pierce, J. Randall, S. Soiland-Reyes, L. Stojanovic, N. Tijanić, Common workflow language, draft 3 (2016).

[39] F. Costa, V. Silva, D. de Oliveira, K. A. C. S. Ocaña, E. S. Ogasawara, J. Dias, M. Mattoso, Capturing and querying workflow runtime provenance with PROV: a practical approach, in: Joint 2013 EDBT/ICDT Conferences, EDBT/ICDT '13, Genoa, Italy, March 22, 2013, Workshop Proceedings, pp. 282–289.

[40] D. Garijo, P. Alper, K. Belhajjame, Ó. Corcho, Y. Gil, C. A. Goble, Common motifs in scientific workflows: An empirical analysis, Future Generation Comp. Syst. 36 (2014) 338–351.

[41] P. Missier, K. Belhajjame, J. Zhao, M. Roos, C. A. Goble, Data lineage model for taverna workflows with lightweight annotation requirements, in: Provenance and Annotation of Data and Processes, Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers, pp. 17–30.

[42] P. Mates, E. Santos, J. Freire, C. T. Silva, Crowdlabs: Social analysis and visualization for the sciences, in: Scientific and Statistical Database Management, Springer, pp. 555–564.

[43] V. Korkhov, D. Krefting, J. Montagnat, T. T. Huu, T. Kukla, G. Terstyanszky, D. Manset, M. Caan, S. Olabarriaga, Shiwa workflow interoperability solutions for neuroimaging data analysis, Stud Health Technol Inform 175 (2012).

[44] B. Ludäscher, I. Altintas, On providing declarative design and programming constructs for scientific workflows based on process networks (2003).

[45] D. Blankenberg, G. Von Kuster, E. Bouvier, D. Baker, E. Afgan, N. Stoler, J. Taylor, A. Nekrutenko, et al., Dissemination of scientific software with galaxy toolshed, Genome Biol 15 (2014) 403.

[46] D. D. Roure, C. A. Goble, R. Stevens, The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows, Future Generation Comp. Syst. 25 (2009) 561–567.

[47] J. Ison, K. Rapacki, H. Ménager, M. Kalaš, E. Rydza, P. Chmura, C. Anthon, N. Beard, K. Berka, D. Bolser, et al., Tools and data services registry: a community effort to document bioinformatics resources, Nucleic acids research 44 (2016) D38–D47.

[48] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, et al., Biocatalogue: a universal catalogue of web services for the life sciences, Nucleic acids research (2010) gkq394.

[49] C. Boettiger, An introduction to docker for reproducible research, ACM SIGOPS Operating Systems Review 49 (2015) 71–79.

[50] F. S. Chirigati, D. Shasha, J. Freire, Reprozip: Using provenance to support computational reproducibility., in: TaPP, International Workshop on Theory and Practice of Provenance.

[51] P. Guo, Cde: a tool for creating portable experimental software packages, Computing in Science & Engineering 14 (2012) 32–35.

[52] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. M. Hettne, R. Palma, E. Mina, Ó. Corcho, J. M. Gómez-Pérez, S. Bechhofer, G. Klyne, C. A. Goble, Using a suite of ontologies for preserving workflow-centric research objects, J. Web Sem. 32 (2015) 16–42.

[53] A. González-Beltrán, E. Maguire, S.-A. Sansone, P. Rocca-Serra, linkedISA: semantic representation of ISA-Tab experimental metadata, BMC Bioinformatics 15 (2014) S4.

[54] G. Juve, A. L. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Comp. Syst. 29 (2013) 682–692.

[55] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, et al., The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud, Nucleic acids research (2013) gkt328.

[56] S. Soiland-Reyes, F. Bacall, P. Hołubowicz, scufl2-wfdesc 0.3. 7 (2014).

[57] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, et al., Galaxy: a platform for interactive large-scale genome analysis, Genome research 15 (2005) 1451–1455.

[58] J. Goecks, A. Nekrutenko, J. Taylor, et al., Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences, Genome Biol 11 (2010) R86.

[59] A. Gonzalez-Beltran, P. Li, J. Zhao, M. S. Avila-Garcia, M. Roos, M. Thompson, E. van der Horst, R. Kaliyaperumal, R. Luo, T.-L. Lee, T.-w. Lam, S. C. Edmunds, S.-A. Sansone, P. Rocca-Serra, From peer-reviewed to peer-reproduced: a role for data standards, models and computational workflows in scholarly publishing, bioRxiv (2014).

[60] C. Pradal, S. Artzet, J. Chopard, D. Dupuis, C. Fournier, M. Mielewczik, V. Nègre, P. Neveu, D. Parigot, P. Valduriez, S. Cohen-Boulakia, Infraphenogrid: A scientific workflow infrastructure for plant phenomics on the grid, Future Generation Computer Systems (2016).

[61] P. D. Tommaso, M. Chatzou, P. P. Baraja, C. Notredame, A novel tool for highly scalable computational pipelines (2014).

[62] D. Koenig, A. Glover, P. King, G. Laforge, J. Skeet, Groovy in action, volume 1, Manning, 2007.

[63] J. P. Kurs, M. Simi, F. Campagne, Nextflowworkbench: Reproducible and reusable workflows for beginners and experts, bioRxiv (2016).

[64] J. Köster, S. Rahmann, Snakemake—a scalable bioinformatics workflow engine, Bioinformatics 28 (2012) 2520–2522.

[65] M. Abouelhoda, S. A. Issa, M. Ghanem, Tavaxy: Integrating taverna and galaxy workflows with cloud computing support, BMC bioinformatics 13 (2012) 1.

[66] O. Biton, S. C. Boulakia, S. B. Davidson, Zoom*userviews: Querying relevant provenance in workflow systems, in: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007, ACM, 2007, pp. 1366–1369.

[67] P. Alper, K. Belhajjame, C. A. Goble, P. Karagoz, Enhancing and abstracting scientific workflow provenance for data publishing, in: Joint 2013 EDBT/ICDT Conferences, EDBT/ICDT '13, Genoa, Italy, March 22, 2013, Workshop Proceedings, ACM, 2013, pp. 313–318.

[68] A. Gaignard, J. Montagnat, B. Gibaud, G. Forestier, T. Glatard, Domain-specific summarization of life-science e-experiments from provenance traces, Web Semantics: Science, Services and Agents on the World Wide Web 29 (2014) 19–30.

[69] L. Hirschman, G. A. C. Burns, M. Krallinger, C. Arighi, K. B. Cohen, A. Valencia, C. H. Wu, A. Chatr-Aryamontri, K. G. Dowell, E. Huala, et al., Text mining for the biocuration workflow, Database 2012 (2012) bas020.

[70] K. M. Hettne, K. Wolstencroft, K. Belhajjame, C. A. Goble, E. Mina, H. Dharuri, D. De Roure, L. Verdes-Montenegro, J. Garrido, M. Roos, Best practices for workflow design: How to prevent workflow decay, in: SWAT4LS.

[71] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (2016).

[72] G. Wilson, Software carpentry: lessons learned., F1000Research 3 (2013) 62–62.

[73] J. Chen, S. Cohen-Boulakia, C. Froidevaux, C. Goble, P. Missier, A. R. Williams, Distillflow: removing redundancy in scientific workflows, in: Proceedings of the 26th International Conference on Scientific and Statistical Database Management, ACM, p. 46.

[74] M. Harmassi, D. Grigori, K. Belhajjame, Mining workflow repositories for improving fragments reuse, in: Semantic Keyword-based Search on Structured Data Sources, Springer, 2015, pp. 76–87.

[75] A. Gibson, M. Gamble, K. Wolstencroft, T. Oinn, C. Goble, K. Belhajjame, P. Missier, The data playground: An intuitive workflow specification environment, Future Generation Computer Systems 25 (2009) 453–459.

[76] B. Ludäscher, I. Altintas, A. Gupta, Compiling abstract scientific workflows into web service workflows, in: Scientific and Statistical Database Management, 2003. 15th International Conference on, IEEE, pp. 251–254.

[77] F. Pérez, B. E. Granger, Ipython: a system for interactive scientific computing, Computing in Science & Engineering 9 (2007) 21–29.

[78] J. S. Racine, Rstudio: A platform-independent ide for r and sweave, Journal of Applied Econometrics 27 (2012) 167–172.

[79] M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, M. Bussonier, The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication., in: AGU Fall Meeting Abstracts, volume 1, p. 07.

[80] F. Chirigati, D. Koop, J. Freire, noworkflow: Capturing and analyzing provenance of scripts, in: Provenance and Annotation of Data and Processes: 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9-13, 2014. Revised Selected Papers, volume 8628, Springer, p. 71.

[81] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. Dey, J. Freire, et al., Yesworkflow: A user-oriented, language-independent tool for recovering workflow information from scripts, arXiv preprint arXiv:1502.02403 (2015).

[82] S. Cohen-Boulakia, U. Leser, Search, adapt, and reuse: the future of scientific workflows, ACM SIGMOD Record 40 (2011) 6–16.

[83] J. Starlinger, B. Brancotte, S. Cohen-Boulakia, U. Leser, Similarity search for scientific workflows, Proceedings of the VLDB Endowment 7 (2014) 1143–1154.

[84] R. Bergmann, Y. Gil, Similarity assessment and efficient retrieval of semantic workflows, Information Systems 40 (2014) 115–127.

[85] Y. Ma, M. Shi, J. Wei, Cost and accuracy aware scientific workflow retrieval based on distance measure, Information Sciences 314 (2015) 1–13.

[86] J. Starlinger, S. Cohen-Boulakia, S. Khanna, S. B. Davidson, U. Leser, Effective and efficient similarity search in scientific workflow repositories, Future Generation Computer Systems 56 (2016) 584–594.