Supporting complex workflows for data-intensive discovery reliably and efficiently

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Khairul Alam

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building, 110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

Scientific workflows have emerged as well-established pillars of large-scale computational science and appeared as torchbearers to formalize and structure a massive amount of complex heterogeneous data and accelerate scientific progress. Scientists of diverse domains can analyze their data by constructing scientific workflows as a useful paradigm to manage complex scientific computations. A workflow can analyze terabyte-scale datasets, contain numerous individual tasks, and coordinate between heterogeneous tasks with the help of scientific workflow management systems (SWfMSs). However, even for expert users, workflow creation is a complex task due to the dramatic growth of tools and data heterogeneity. Scientists are now more willing to publicly share scientific datasets and analysis pipelines in the interest of open science. As sharing of research data and resources increases in scientific communities, scientists can reuse existing workflows shared in several workflow repositories. Unfortunately, several challenges can prevent scientists from reusing those workflows, which hurts the purpose of the community-oriented knowledge base. In this thesis, we first identify the repositories that scientists use to share and reuse scientific workflows. Among several repositories, we find Galaxy repositories have numerous workflows, and Galaxy is the mostly used SWfMS. After selecting the Galaxy repositories, we attempt to explore the workflows and encounter several challenges in reusing them. We classify the reusability status (reusable/nonreusable). Based on the effort level, we further categorize the reusable workflows (reusable without modification, easily reusable, moderately difficult to reuse, and difficult to reuse). Upon failure, we record the associated challenges that prevent reusability. We also list the actions upon success. The challenges preventing reusability include tool upgrading, tool support unavailability, design flaws, incomplete workflows, failure to load a workflow, etc. We need to perform several actions to overcome the challenges. The actions include identifying proper input datasets, updating/upgrading tools, finding alternative tools support for obsolete tools, debugging to find the issue creating tools and connections and solving them, modifying tools connections, etc. Such challenges and our action list offer guidelines to future workflow composers to create better workflows with enhanced reusability. A SWfMS stores provenance data at different phases of a workflow life cycle, which can help workflow construction. This provenance data allows reproducibility and knowledge reuse in the scientific community. But, this provenance information is usually many times larger than the workflow and input data, and managing provenance data is growing in complexity with large-scale applications. In our second study, we document the challenges of provenance management and reuse in e-science, focusing primarily on scientific workflow approaches by exploring different SWfMSs and provenance management systems. We also investigate the ways to overcome the challenges. Creating a workflow is difficult but essential for data-intensive complex analysis, and the existing workflows have several challenges to be reused, so in our third study, we build a recommendation system to recommend tool(s) using machine learning approaches to help scientists create optimal, error-free, and efficient workflows by using existing reusable workflows in Galaxy workflow repositories. The findings from our studies and proposed techniques have the potential to simplify the data-intensive analysis, ensuring reliability and efficiency.

Acknowledgements

At first, I like to praise Almighty God, the most gracious and most merciful, who gave me the ability to carry out this work. Next, I would like to express my sincerest appreciation to my supervisor Dr. Banani Roy for her continuous support, guidance, motivation, and remarkable endurance during this thesis work. Without her support, this work would have been unthinkable.

I would like to thank Dr. Ian Stavness, and Dr. Derek Eager for their willingness to take part in the advisement and evaluation of my thesis work. I would also like to thank them for their valuable time.

I would like to express my special appreciation to Dr. Chanchal K. Roy for his guidance, valuable suggestions, and comments on improving this thesis.

I would also wish to express my gratefulness to Dr. Alexander Serebrenik, Professor, Eindhoven University of Technology, for extended discussions, valuable suggestions, passionate participation, and input, which have contributed significantly to the improvement of the thesis.

I express my heartiest gratitude to my sister Reshma Khatun and my mother, Mst Nasima Begum, who are the architects of my life. Their endless sacrifice, unconditional love, and constant good wishes have made me reach this stage of my life. I am also thankful to my Father, Md Moin Uddin Shaikh, and other family members for their support and inspiration in my bad times.

I would like to convey my greatest respect and heartiest gratitude to my beloved friend Shamimur Rahman and Abdul Awal for helping me to see the new light of life and start anew.

I am very grateful to my lovely wife, Tasnim Islam Mou. She sacrificed a lot for me to obtain a higher education. I am also thankful to my son Rufaid Fayaz Tazin.

Thanks to all of the members of the Interactive Software Engineering and Analytics (iSEA) Lab and the Software Research (SR) Lab, with whom I have had the opportunity to grow as a researcher. In particular, I would like to thank Saikat Mondal, Amit Kumar Mondal, Muhammad Mainul Hossain, Tonny Kar, Debasish Chakroborti, Sristy Sumana Nath, Saumendu Roy, Subrato Nag, Md Nadim, Shamse Tasnim Cynthia, Rayhan Islam Shuvo, Jarin Tasnim, Palash Ranjan Roy, Ajmain Inqiad Alam and Eslamloo Parnian.

Most importantly, I am grateful to Natural Sciences and Engineering Research Council of Canada (NSERC), Collaborative Research and Training Experience (CREATE), and University of Saskatchewan for supporting my study through NSERC Industry Engage and Discovery grants, and Graduate Teaching Fellowships (USask).

I would like to thank all of my friends and other staff members of the Department of Computer Science who have helped me to reach this stage. In particular, I would like to thank Raouf Ajami, Greg Oster, Maurine Powell, Cary Bernath, Smit Choksi, Sophie Findlay, Shakiba Jalal, Heather Webb, Cynthia Kovacs and James Ko.

I dedicate this thesis to my sister, Reshma Khatun. Her priceless love, courage, and inspiration give me strength and hope to be a well-educated and good human being. I also want to dedicate this thesis to my wife, Tasnim Islam Mou, and my Son, Rufaid Fayaz Tazin. They all sacrificed a lot for me to obtain this degree.

Contents

List of Tables

List of Figures

List of Abbreviations

SWfMSs	Scientific Workflow Management Systems
SW	Scientific Workflow
SWR	Scientific Workflow Repositories
DAG	Directed Acyclic Graph
LOF	List of Figures
LOT	List of Tables
ML	Machine Learning
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DT	Decision Tree
GRU	Gated Recurrent Units
GPU	Graphics Processing Unit
WMS	Workflow Management System
RWM	Reusable Without Modification
ER	Easily Reusable
MDR	Moderately Difficult to Reuse
DR	Difficult to Reuse
NR	Nonreusable
SQL	Structured Query Language

1 Introduction

1.1 Motivation

Scientific workflows and Scientific Workflow Management Systems (SWfMSs) have become excellent mediums for analyzing large-scale, heterogeneous, and complex scientific data. Scientific Workflows provide the abstractions and mechanisms to manage complex computational tasks across computing resources for diverse domains. A common strategy to solve complex silico experiments is to model them as workflows and to use a workflow management system to organize their execution [107]. The importance of workflows is highlighted as they have underpinned some of the most significant discoveries *(life science, earth science, chemistry, astrophysics, and so on)* of the past decades. With the increasing popularity of scientific workflows in the scientific community, public workflow repositories are gaining importance as a means to share, collaborate and find workflows. They are favorable in terms of reusability, as previously designed workflows can be made publicly available through several workflow repositories and then can be used in other workflows [88]. Despite the impressive results to date, workflow construction, research, and development are still ad-hoc [29]. Studies show that constructing scientific workflows is very complex and challenging for new as well as expert users due to the significant growth of tools, the heterogeneous nature of data, and the complexity of the tasks.

Some popular SWfMSs are Galaxy [81], Taverna [137], Wings [79], Vistrails [28], Pegasus [50], Askalon [58], Kepler [4], Askalon [59], VizSciFlow [93], SciWorCS [131], VIEW [110], Triana [157], CoGe [118] and Knime [163]. Some popular workflow repositories are Crowdlabls [123], WorkflowHub [45], DockStore [136], and Galaxy workflow repositories [75, 70, 66, 69, 80, 74, 65]. Although several SWfMSs have been developed, and there are several workflow repositories for sharing and reusing workflows, there exist several challenges preventing scientists from reusing a workflow. Finding these challenges and overcoming them can assist scientists in creating more effective, error-free, and optimal workflows, which will make complex data analysis more reliable, efficient, and simple.

A SWfMS automates a workflow using different phases (*composition, deployment, execution, and analysis*) of a scientific workflow life cycle. The execution of workflows is costly and requires a lot of resource usage. At different phases of a workflow life cycle, most SWfMSs store provenance information (discussed in detail in Chapter 2), allowing result reproducibility, sharing, and knowledge reuse in the scientific community. But, this provenance information can be many times larger than the workflow and input data [43, 149], and managing provenance data is growing in complexity with large-scale applications. Handling exponentially increasing data volume and utilizing the technical resources for storage and computing are thus demanded by

exploiting data-intensive computing in various application fields. There are several studies [31, 6, 122, 108, 102, 87, 86, 9, 138, 142, 27] about storing, querying provenance information, audit trails using provenance also reproducibility issues of a scientific workflow using provenance data. Workflow provenance assures the reliability and integrity of workflows and the data as they are routed in complex workflows. There have several drawbacks to storing and reusing provenance information. Identifying and resolving these problems will save storage, execution time and make the systems efficient to use.

The task of a recommender system is to suggest relevant items to users based on their preferences and behavior [117]. This involves analyzing data such as user ratings, past purchases, browsing history, and demographic information to generate personalized recommendations. The ultimate goal is to improve the user's overall experience, increase engagement, and drive sales or user satisfaction. Recommender systems have been used in online shopping, travel booking, and media service providers for many years. It is also used extensively in the scientific community for scientific literature searches to help scientists explore relevant and recent papers quickly. Commercial companies like Alibaba and Amazon use their customers' preferences to select products from an extensive collection. Production companies like YouTube and Netflix suggest new songs, movies, etc., based on the previous usage of the customers. In short, recommender systems make life smoother for the users to find appropriate things among the ocean of products. Recommender systems differ in how they analyze data sources to develop notions of affinity between users and items, which can be used to identify well-matched pairs [127]. The successful usage of recommendation systems by companies to find relevant things encourage us to create a tool(s) recommendation system to help scientists create optimal, error-free, and efficient workflows by analyzing existing workflows in various workflow repositories. The tool(s) recommendation system can save time researchers waste in creating erroneous or less optimal workflows by choosing improper tools which produce unexpected results. It will also relieve researchers from memorizing a vast amount of tools and increase the accessibility of the tools. In addition, the recommended tools can facilitate further data analysis by recommending high-quality and highly used tools.

1.2 Problem Statement

1. **Non-reusable Scientific Workflows in Workflow Repositories:** A variety of workflow systems are in use in a variety of scientific domains, such as genomics, bioinformatics, astronomy, cheminformatics, etc. A workflow can be considered a software artifact; scientists can share and exchange it once developed and tested. Other scientists can reuse existing workflows in their analysis, e.g., as sub-workflows [171]. Scientific workflow repositories are gaining popularity and starting to emerge as scientists can share, find and reuse such workflows. A number of studies [156, 154, 53, 38] have been done to explore these repositories, finding similarities among workflows, mining usages patterns, adapting workflows, and so on. Unfortunately, a significant percentage of workflows from repositories [132, 167, 56, 75, 70, 66] are nonreusable. But according to FAIR (*Findable, Accessible, Interoperable and Reusable*) principles [164], metadata and data should be well-

described so that they can be reused in different settings. Unfortunately, there is a marked lack of research investigating the challenges of reusing workflows and how to overcome these challenges. Finding the reusability challenges and overcoming them can guide the workflow developers to construct workflow more effectively.

2. **Workflows Composition is Difficult, and Repetitive Executions are Costly:** While executing workflows, SWfMSs generate provenance data which is essential for result reproducibility, audit trail, sharing, and knowledge reuse in the scientific community [26]. Recent efforts from the scientific workflow community aiming at largescale capturing provenance present a new opportunity for using provenance to provide recommendations during building scientific workflows [170]. Provenance data generation is mostly depended on how a workflow is composed, and the cost of managing provenance is also directly related to the workflow construction. Most of the SWfMSs support workflow provenance to assure the reliability and integrity of workflows. However, systematically capturing and managing provenance information for computational tasks has recently received significant attention for its relevance to various domains and applications [61]. There have been many provenance models for managing provenance data. They have reproducibility facilities that are important with respect to the scientific field, and lack of it can be a burden to an individual researcher [148]. But if a workflow needs to be executed several times, there are better ways to manage data instead of reproducing the same thing all the time. However, while executing workflows in SWfMSs, these models do not reuse previously stored data for re-execution. Also, there are many drawbacks to storing provenance data. So finding out the challenges of provenance management, reuse, and storing is still an open research problem and requires further investigation. Finding the challenges and overcoming them can lead to better provenance management, and it can also be used to build [170] scientific workflows.

3. **Lack of Proper Tool(s) Recommendation System to Construct a Workflow:** Software developers and workers from other technical domains spend a significant fraction of their working time searching for information, for example, to understand existing code, to select appropriate methods to implement a feature, or to find proper tools and techniques to solve a particular problem [145]. Recommendation systems in various sectors like software engineering, e-commerce, scientific literature search, e-learning, and so on asses the needs of their respective users in suggesting relevant items. Several SWfMSs such as Bcbio-nextgen, Omics Pip, Nextflow, Luigi, Toil, and so on also have recommendation facilities [106]. All these approaches are restricted to a few workflow generations. Also, they depend on either annotations or matching input and output data types of adjacent tools in workflows. There are also some systems [104, 106] which suggest tools based on previous workflows. But these approaches have several challenges, and that's why a significant percentage of workflows in workflow repositories have reusability issues. So, considering them a tool(s) recommendation system to construct a scientific workflow requires further investigation.

The research problems we attempt to solve in this thesis can help domain scientists analyze their complex and heterogeneous data effectively and efficiently.

1.3 Research Contributions

Focusing on the above research issues scientists face using SWfMSs, our studies make the following three major contributions. Here in this section, we briefly present our contributions to the study.

1.3.1 Study 1: Reusability Challenges of Scientific Workflows: A Case Study for Galaxy

In this study, we report an exploratory study on the reusability challenges in 307 workflows related to metagenomics, virology, RNA sequencing, CHIP sequencing, etc., from the most popular SWfMS, Galaxy. In particular, we import, compile, execute, and even carefully examine the annotations and tools parameters and then attempt to reuse the workflows. Several challenges prevent us from reusing a significant number of workflows. Then we try to resolve the challenges, and in some cases, we become successful, and we fail in other cases. We find that 26.71% of workflows are nonreusable even after trying. On the other hand, we can reuse 30.94% of workflows by doing some major/minor modifications, and 42.35% of workflows are reusable without any changes. We also carefully investigate the challenges and find ways to overcome them.

1.3.2 Study 2: Challenges of Provenance in Scientific Workflow Management Systems

In this study, we explore state-of-the-art provenance management mechanisms. In different phases of a workflow life cycle, most SWfMSs capture provenance information, and their capturing process differs from one to another. A proper record of provenance information is vital in many scientific analyses. It helps to verify and reproduce the results of any investigation. A workflow may contain numerous individual tasks, and many of them may take from hours to days to complete the tasks. Scientists often need to execute a workflow module or sub-workflow multiple times to do an analysis. If we can reuse the previously stored provenance data for workflow re-execution, then it can help reduce the cost of storage and execution time. This study documents the challenges of provenance management and reuses in e-science, focusing primarily on scientific workflow approaches by exploring different SWfMSs and provenance management systems. We also investigate the ways to overcome the challenges.

1.3.3 Study 3: Recommending Tool(s) for Scientific Workflow Management Systems

In this study, we offer a tool(s) recommendation mechanism for constructing scientific workflows. Constructing scientific workflows is a very complex task, and our system will help to reduce the complexity by suggesting relevant tool(s). Here, we first collect the workflows (reusable/we make them reusable by performing actions)

from various workflow repositories for Galaxy SWfMS and then build a model to suggest tool(s) for constructing a workflow. While building the model, we take into consideration the tools' usage frequencies, period, workflows annotations, workflows tool sequences, and workflow correctness. Our model can help workflow designers save time they invested in searching or memorizing tools for constructing workflows.

1.4 Publications

Following is a list of published, under review, and prepared papers for submission (with co-authors) from this thesis.

- Khairul Alam, and Banani Roy. Challenges of Provenance in Scientific Workflow Management Systems. Paper accepted at *Workflows in Support of Large-Scale Science (WORKS), Dallas, Texas, United States. 2022*
- Khairul Alam, Banani Roy, and Alexander Serebrenik. Recommending Tools and Sub-workflows for Scientific Workflow Management Systems. Paper accepted at *Workflows in Support of Large-Scale Science (WORKS), Dallas, Texas, United States. 2022*
- Khairul Alam, Banani Roy, and Alexander Serebrenik. Reusability Challenges of Scientific Workflows. Paper under review at *20th International Conference on Mining Software Repositories (MSR) 2023*
- Khairul Alam, and Banani Roy. Exploring and Mining of Scientific Workflow Repositories. *International Conference on Empirical Software Engineering and Measurement (ESEM-2023)* (to be submitted)

1.5 Thesis Outline

The thesis has six chapters in total. We conduct three independent but interrelated studies to determine the challenges scientists face using scientific workflows and SWfMSs and assist data-intensive analysis. This section outlines the chapters of the thesis.

- Chapter 2 discusses the background concepts of this thesis, such as Scientific Workflows, SWfMSs, Scientific Workflow Repositories, Provenance, Machine Learning Algorithms, and so on.
- Chapter 3 describes the first study we used to discover the reusability challenges of scientific workflows from Galaxy workflow repository and guides scientists to create error-free and effective workflows.
- Chapter 4 discusses the challenges of managing provenance information using current SWfMSs and provenance models, and later we propose approaches for optimal storage and reuse of provenance data.
- Chapter 5 presents our tool(s) recommendation system to recommend tool(s) to help scientists construct scientific workflows without facing complexity.

- Chapter 6 concludes the thesis with the overall summary and discussion of some future research directions.

2 Background

In this chapter, we briefly discuss the background and technical preliminaries of the thesis. In Section 2.1, we first provide a discussion on Scientific Workflows. Section 2.2 and Section 2.3 describe the Scientific Workflow Life Cycle and Scientific Workflow Repositories. Section 2.4 discusses about Scientific Workflow Management Systems (SWfMSs). Then we discuss the relationships among Software Engineering, Scientific Workflows, and SWfMSs in Section 2.6. Section 2.7 illustrates the fundamental idea of prospective and retrospective provenances. Finally, in Section 2.8, we briefly describe various Machine Learning Algorithms that we use in this thesis.

2.1 Scientific Workflows

Modern scientific collaborations have opened the opportunity to solve complex problems with heterogeneous data requiring multidisciplinary expertise and large-scale computational experiments. These experiments usually consist of a sequence of processing steps in a computing platform. An efficient way to manage these experiments is to model them as workflows. A workflow automates a process by executing different logical data processing activities according to predefined rules. Workflows can be divided into *Business Workflows* and *Scientific Workflows*. A business workflow automates a business process in whole or part according to a set of procedural rules [10], which eventually makes business processes more efficient and reliable. Scientific workflows are typically used for modelling and running scientific experiments [112]. A scientific workflow assembles complex sets of scientific data processing activities with data dependencies among them. In addition, a scientific workflow can contain one or more sub-workflows composed of a subset of activities and data dependencies in the scientific workflow. We can represent a scientific workflow as a directed acyclic graph (DAG)[1] in which nodes correspond to data processing activities, and edges represent the data dependencies between nodes.

A scientific workflow has inputs and outputs, data processing modules/pipelines which operate on data. The datalinks connect the former and thereby define the data flow from one module to the following modules. Workflow developers can give a name to a workflow. It can also have an annotation. Each workflow module has attributes associated with it, such as a label, input, and output, the type of operation to be conducted, parameters, etc.

Lin et al. [110] define scientific workflow as computerized facilitation or automation of a scientific process,

[1] (https://cran.r-project.org/web/packages/ggdag/vignettes/intro-to-dags.html)

in whole or part, which usually streamlines a collection of scientific tasks with data channels and dataflow constructs to automate data computation and analysis to enable and accelerate scientific discovery.

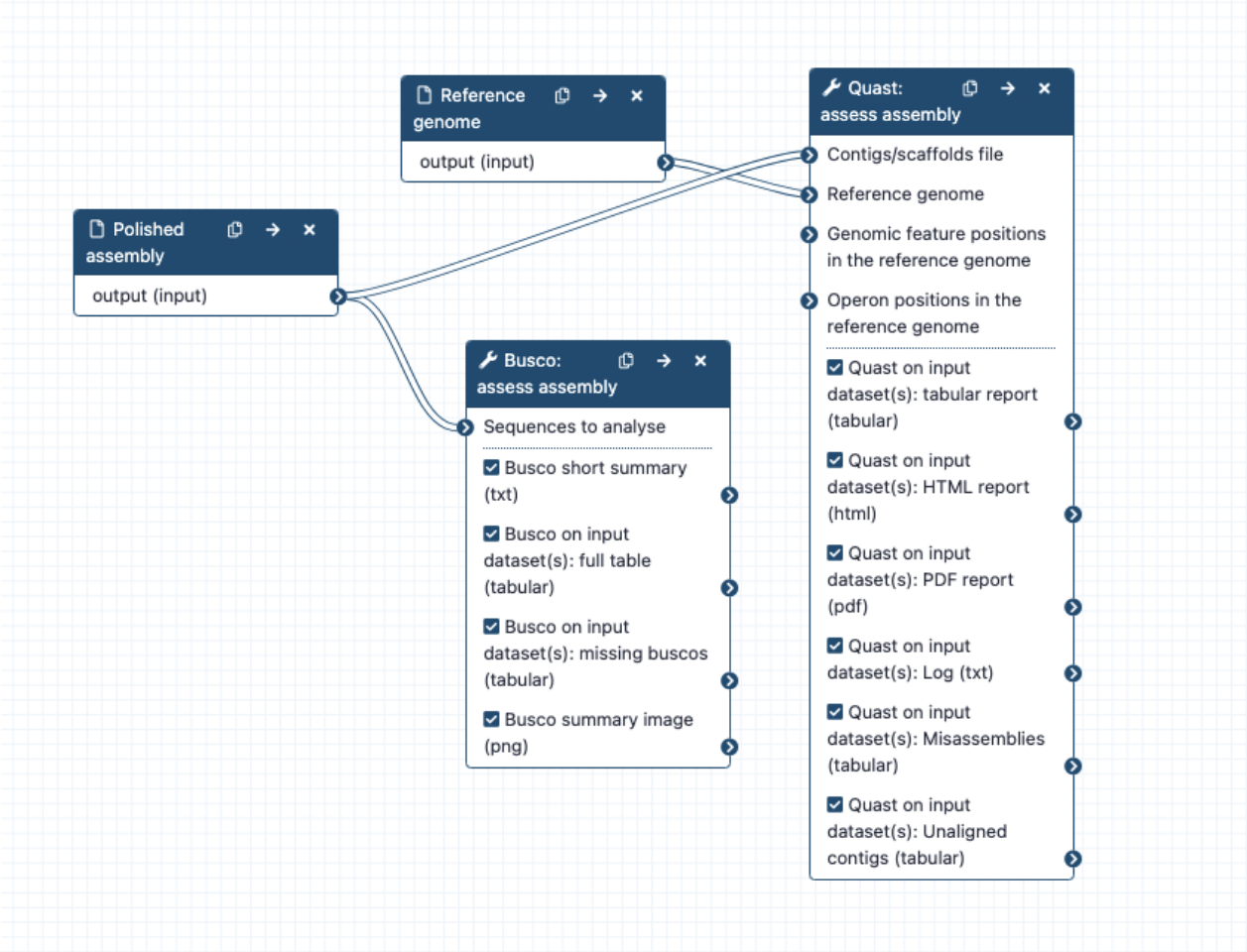

Figure 2.1: An example of a scientific workflow (Adapted from WorkflowHub [167])

Figure 2.1 shows an example of a scientific workflow obtained from WorkflowHub [167]. This workflow assesses genome quality by generating some statistics and determining if expected genes are present, aligning contigs to a reference genome. The inputs of the workflow are polished assembly and reference_genome.fasta. Here, two software tools are *Busco, and Quast* used. The outputs contain Busco table of genes, Quast HTML report, etc. This workflow can run alone or as part of a combined workflow for large genome assembly.

The workflow steps are known as *Computational Modules*. The computational modules are responsible for data manipulation and processing. A workflow module starts its execution when the required datasets are available.

2.1.1 Scientific vs. business workflows

Scientific and business workflows have overlapping properties as they began from the same common ground but have several differences. For example, a business workflow tools look more like traditional programming

languages. In contrast, a scientific workflow exploits higher abstraction-level tools to plug together problem-solving components to prove a scientific hypothesis. Secondly, in business workflows, data can be processed by different participants (both machines and humans), but in scientific workflows, data is processed only by machines; scientists can monitor or control the execution [113]. Thirdly, the validation of a scientific hypothesis depends on experimental data, so a scientific workflow tends to have an execution model that is dataflow oriented. In contrast, a business workflow emphasizes control-flow patterns and events [10]. The scientific workflow construction uses an incremental approach, but a workflow will be designed first for business workflow, then implementation occurs. Also, the purpose of business workflows is to reduce human resources (and other costs) and increase revenue. On the other hand, scientific workflows aim to reduce both human and computation costs and accelerate the speed of turning large amounts of bits and bytes into knowledge and discovery. Finally, scientific workflows must be fully reproducible, which is optional for business workflows [10]. Hereafter, a workflow indicates a scientific workflow.

2.2 Scientific Workflow Life Cycle

The life cycle of a scientific workflow describes the state transitions of a scientific workflow from creation to completion. According to Gorlach et al. [84], a scientific workflow life cycle contains four phases: modelling phase, deployment phase, execution and monitoring phase, and analysis phase. In addition, Mattoso et al. [43] incorporate provenance issues (Figure 2.2) with the scientific workflow life cycle in silico experiments and silico simulations.

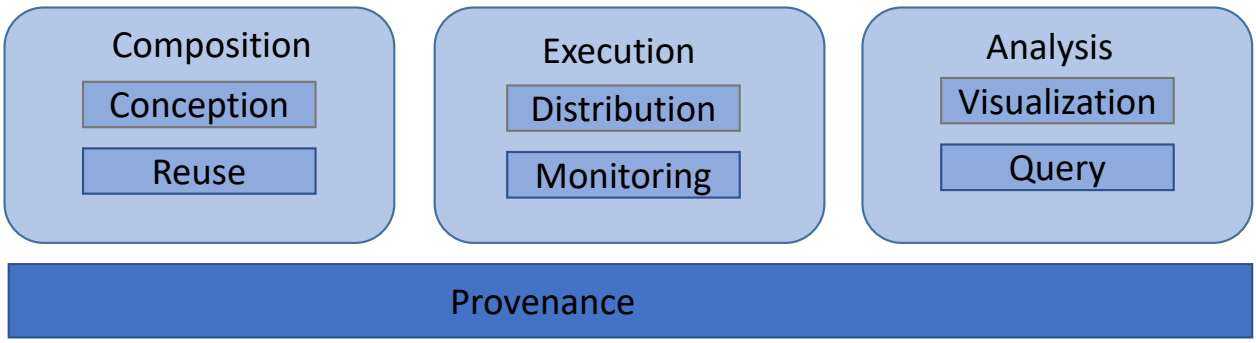

Figure 2.2: The life cycle of a scientific workflow (Adapted from [43])

By combining several scientific workflow life cycle views [84, 51, 124], Liu et al. [113] finalize four phases in a scientific workflow life cycle.

- **Composition Phase:** Scientists define the experiment's hypothesis and select partners, resources, and data in this phase. SWfMSs have the facilities for composing workflow textually or by using Graphical User Interface (GUI). During composition, scientists can add data activities or control flow structures to the workflow. In addition, SWfMSs users can reuse the existing shared scientific workflows with or

without modifications [90].

- **Deployment Phase:** The deployment phase in a scientific workflow involves the implementation of the scientific solution or model developed in earlier phases. This phase typically involves deploying the solution to a production environment or integrating it with an existing system or workflow. Deployment often requires establishing connections with data sources, storage, and processing resources. This phase may also involve performing final testing, debugging, and optimization of the solution to ensure it runs efficiently and produces accurate results. Sometimes, the deployment phase may also involve training end-users or providing technical support.

- **Execution Phase:** The execution phase in a scientific workflow involves implementing the steps and processes defined in earlier phases to collect, process, analyze, and interpret data. Researchers typically run their experiments or simulations in this phase, perform data analyses or modeling, and generate results or outcomes. Execution often involves using software tools, algorithms, or computational methods to process and analyze data at scale. This phase may also involve implementing quality control measures to ensure the integrity and reproducibility of the results. The execution phase is typically iterative, with researchers refining their methods and approaches as they collect and analyze more data. The results of the execution phase are often used to inform the next steps in the scientific workflow, such as refining the research question or hypothesis, modifying the experimental design, or developing new models.

- **Analysis Phase:** The analysis phase of a scientific workflow involves examining and interpreting the data collected in earlier phases in order to answer research questions and test hypotheses. In this phase, researchers typically perform statistical analyses, apply modeling techniques, and visualize data in order to understand relationships, patterns, and trends in the data. The analysis phase often involves identifying and filtering out data that is irrelevant, inconsistent, or unreliable and using quality control measures to ensure that the results are accurate and reproducible. The results of the analysis phase are typically used to draw conclusions and develop insights that can be communicated to stakeholders or used to inform further research. The analysis phase can also be iterative, with researchers refining their analyses and hypotheses as they gain new insights from the data.

2.3 Scientific Workflow Repositories

Several challenges prevent easy adoption, maintenance, and consistency with production workflows' evolving structures and computational requirements. So researchers developed several community frameworks for analyzing workflows. Furthermore, scientific workflows often are complex, so sharing them employing public workflow repositories has become an essential issue for the community. As a result, public repositories are gaining importance as a means to share, find, and reuse such workflows. Moreover, open science has emerged

as a framework for improving the quality of scientific analysis. Transparent, accessible knowledge-sharing, and collaborative networks are essential components of open science. Scientific workflow communities are also tending toward open science and, as a result, made many workflows available to the community in different repositories, i.e., WorkflowHub [45], Galaxy [75] and myExperiment [80]. Some other popular workflow repositories are [167, 70, 66, 69, 132, 56, 74, 65].

2.4 Scientific Workflow Management Systems

A Scientific Workflow Management System (SWfMS) is a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or workflow, in a scientific application [8]. It is a powerful tool to execute workflows in a scientific workflow engine, a software service that provides the runtime environment for workflow execution. Lin et al. [110] define SWfMS as a system that completely defines, modifies, manages, monitors, and executes scientific workflows through the execution of scientific tasks whose execution order is driven by a computerized representation of the workflow logic. To execute a scientific workflow in a given environment, a SWfMS typically generates a Workflow Execution Plan (WEP), a program that captures optimization decisions and execution directives, typically the result of compiling and optimizing a workflow before execution. Most of the SWfMSs support provenance (metadata that captures the derivation history of a dataset, including the original data sources, intermediate datasets, and the workflow computational steps), which is used for workflow analysis and workflow reproducibility [113].

Pegasus and Swift can work with terabytes of data with excellent support for scalability and high performance. Kepler, Taverna, and Triana have a graphical user interface for desktop computers. For algebraic operation, Chiron performed best to the existing SWfMSs. Galaxy has a GUI that users can access through web browsers. On the other hand, Askalon implements desktop and web GUI adapted to cloud environments. Pegasus, Swift, and Chiron design and execute a workflow through a textual interface, while Kepler, Taverna, Galaxy, Triana, and Askalon integrate a GUI for workflow design. Galaxy works with bioinformatics workflows, and Pegasus, Kepler, Swift, and Taverna are mostly used in astronomy, biology, and other domains. We are providing some key features of some mostly used SWfMSs.

- **Galaxy**:
    1. Galaxy is a web-based SWFMS for genomic research.
    2. It provides a GUI for designing scientific workflows through browsers.
    3. Users can upload data, share workflow information, import workflows from myExperiment.
    4. Galaxy generates concrete tasks for each activity, puts the tasks in a queue to be submitted, and monitors the task status (in queue, running, or completion).
    5. Tasks are executed using the greedy approach.

- **Pegasus**:

    1. Widely used in astronomy, bioinformatics, climate modeling, earthquake science, and genome analysis.
    2. Support provenance gathering and querying through Pegasus/Wings framework. The provenance data or monitoring data come from the log data gathered during workflow execution.
    3. Key features are portability in different infrastructures such as grid and cloud, optimized scheduling algorithm, good scalability, support for provenance data that can be used for debugging, data transfer support for data-intensive workflows, fault-tolerance support, detailed user guide and available package in Debian repository.
    4. Check available intermediate data in the available computing nodes for reducing the workflow computational steps.
    5. Pegasus hides the complex scheduling, optimization, and data transmission of workflows from SWfMS users.

- **Swift**:

    1. Swift has been used in biology, astronomy, economics, and neuroscience.
    2. Executes data-intensive scientific workflows through five functional phases: Program specification, scheduling, execution, provenance management, and provisioning.
    3. Swift grew out of the GriPhyN Virtual Data System (VDS), whose objective is to express, execute, and track the results of workflows through program optimization and scheduling, task management, and data management.
    4. Provenance data is available in the user service layer.
    5. Swift achieves fault tolerance by retrying the failed tasks and providing a restart log when the failures are permanent.

- **Kepler**:

    1. Kepler is used in oceanography, data management, and biology.
    2. Kepler is built upon the Ptolemy II system.
    3. Kepler integrates a powerful graphical workbench.
    4. Provenance functionality is managed by the provenance recorder (PR).

- **Taverna**:

    1. Taverna is used in astronomy, bioinformatics, and chemistry.
    2. GUI for designing workflows and textual language to represent a workflow as a DAG.

3. Workflow can be shared through myExperiment.

4. Taverna is able to use the computational resources from the grid or cloud.

2.4.1 Reference architecture for SWfMSs

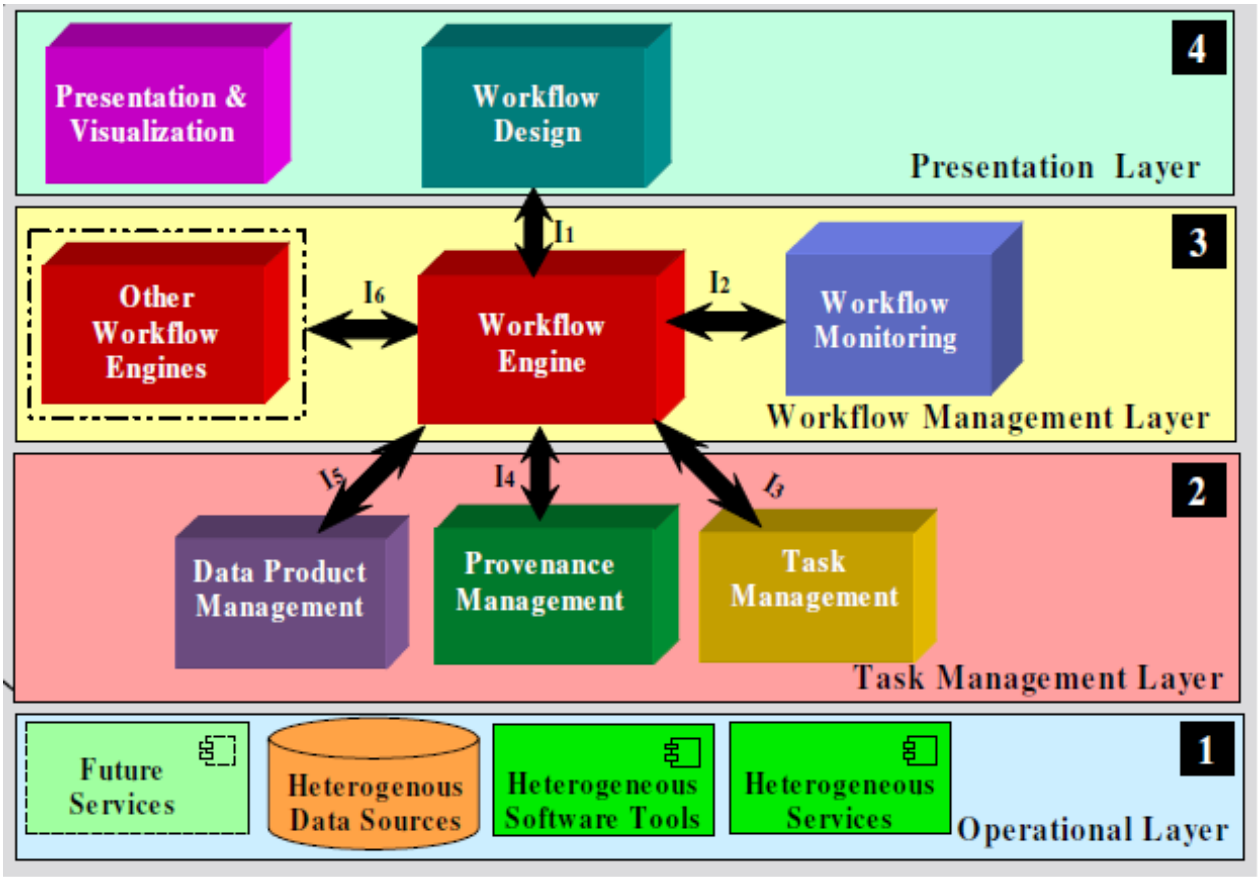

Figure 2.3: Reference architecture for SWfMSs (Adapted from [110])

Lin et al. [110] proposed a four layers reference architecture for SWfMSs. Figure 2.3 shows the reference architecture. The first layer is the Operational Layer, which consists of a wide range of heterogeneous and distributed data sources, software tools, services, and their operational environments, including high-end computing environments [110]. The second layer is called the Task Management Layer. Tasks are the building blocks of scientific workflows. Tasks consume input data products and produce output data products. At the same time, provenance is captured automatically to record the derivation history of a data product, including original data sources, intermediate data products, and the steps that are applied to produce the data product [110].

The third layer is the Workflow Management Layer, which is responsible for executing and monitoring scientific workflows. At this layer, the building blocks of a scientific workflow are the tasks provided by the underlying Task Management Layer [110]. The interaction between users and SWfMS at all stages of a

scientific workflow life cycle is performed in the presentation layer. A workflow's designing, assembling, and data processing activities are done here. The functionality of showing execution status is also shown here. Cui Lin et al. [110] present seven key architectural requirements for a SWfMS discussed as follows:

- **User interface customizability and user interaction support:**

  In a SWfMS, users design, modify, execute, and monitor scientific workflows. Graphical user interfaces are difficult to use for a large workflow; in that case, domain-specific visualization is essential. Users should have the flexibility of customizing the user interface based on different scientific domains and engineering disciplines. Customizing the user interface should be on the user, and it should not affect any other functional components of the system.

- **Reproducibility support:**

  A result obtained from the execution of a workflow must be reproducible. So sufficient provenance information capturing, including derivation history, needs to be maintained. A key functional component for a SWfMS is the management of provenance metadata, from the collection, representation, storage, and querying to visualization.

- **Heterogeneous and distributed services and software tools integration:**

  In recent days, data has become very complex as well as massive. Scientists need to work with a range of heterogeneous analytical and computational services and various software tools to execute a particular workflow. So an ideal SWfMS should have a tools integration facility.

- **Heterogeneous and distributed data product management:**

  The execution of a scientific workflow sometimes generates a huge volume of distributed data objects. These data objects can be primitive or complex. These heterogeneous data can be overwhelming for scientists. So a SWfMS should have the efficient management of data products.

- **High-end computing support:**

  An ideal SWfMS should have grid computing and cloud computing facilities.

- **Workflow monitoring and failure handling:**

  For long-running scientific queries, monitoring workflow execution progress is essential, and the system should not have a single point of failure.

- **Interoperability:**

  Many scientific research projects are collaborative in nature and involve multiple geographically distributed organizations, and sub-workflows may run in different SWfMSs. Therefore, a SWfMS can take advantage of other SWfMSs.

2.5 Scientific Workflow Examples

Many workflow users are reluctant to release their code and data. To date, the community has lacked detailed knowledge of a range of scientific workflows [99], but several workflows are currently available. NASA/IPAC [2] created Montage astronomy workflow [15] for the Pegasus SWfMS as an open-source toolkit; it can be executed in grid environments and is used to evaluate workflow algorithms. SciEvol [135] is executed in the Chiron SWfMS. In the bioinformatics domain, SciEvol is a workflow for molecular evolution reconstruction. Some data-intensive workflows in bioinformatics are SciPhylomics [49], SciPPGx [134], SciPhy [133]. All these workflows are executed in SciCumulus SWfMS [48]. There are also some other available workflows like Cybershake [119], Broadband [30], LIGO inspiral analysis workflow [1], Coronavirus sequencing [140], and SIPHT [114]. Scientists can publish their workflows in WorkflowHub[3], which is a collaborative environment. The other repositories are [75, 70, 66, 56], and so on. Scientists share their workflows with these repositories so that other users can reuse them to solve their problems.

2.6 Software Engineering, Scientific Workflows and SWfMSs

The increasing complexity of data and analysis methods has created such an environment where domain scientists, who may not know about software engineering, are finding themselves playing the impromptu role of software engineer through the use of scientific workflows and SWfMSs. We can consider a scientific workflow as software typically used to solve computational problems. Like algorithms, scientific workflows are used for data processing and performing calculations. SWfMSs are like IDEs (Integrated Development Environments), such as Eclipse or PyCharm, for creating and executing a scientific workflow. While developing software, we use classes and methods for completing tasks. In the same way, while constructing workflows, we use module pipelines to accomplish a task. We have control flow and data flow facilities in scientific workflows like software development. Scientific workflows and SWfMSs are usually for domain scientists with little or no programming knowledge. Still, the purpose of software engineering and scientific workflows has similarities in order to take a scientific workflow to the production level; the scientists also need to follow the software engineering life cycle, which includes requirements gathering, designing, implementation, testing, evolution, and maintenance. Along with bioinformatics, chemistry, pharmacology, and climate analysis tools, Galaxy SWfMS has numerous statistics and machine learning tools through which software developers and domain scientists can benefit. We can consider the SWfMSs like Galaxy, Taverna, LONI pipeline, or Generation as an integrated software framework for performing scientific data analysis.

[2] http://montage.ipac.caltech.edu/
[3] https://workflowhub.eu/

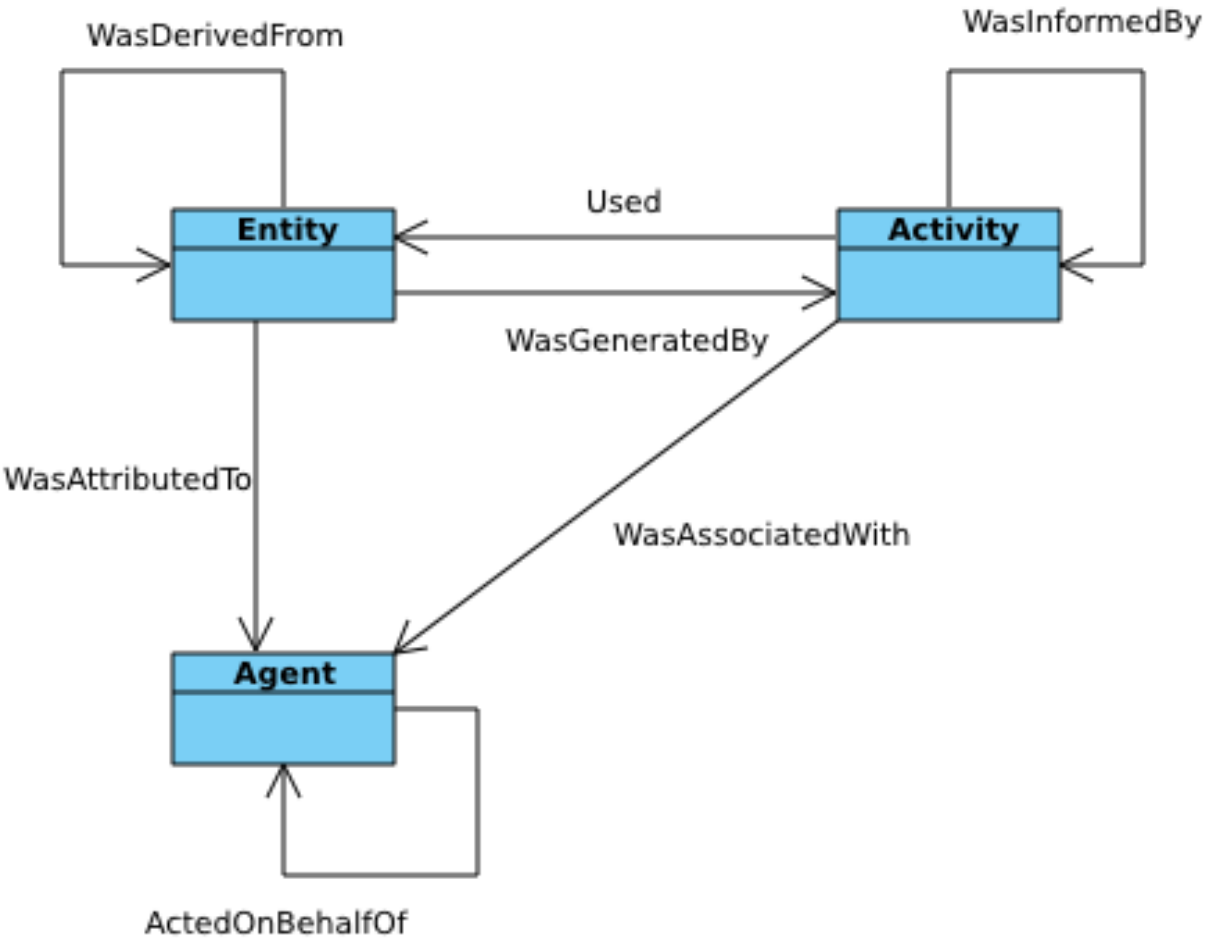

Figure 2.4: Provenance core structure (Adapted from [161])

2.7 Provenance

Provenance was mostly used in relation to works of art, but it is now extensively used in a wide range of fields, especially in science and computing. The primary purpose of provenance is to provide contextual and circumstantial evidence for its discovery. The term provenance is conceptually similar to the term *chain of custody*. Some other terms related to provenance are *traceability, lineage, logging, and monitoring*. It covers the complete documented history of an object. The provenance in a scientific workflow is essential for its ability to reproduce the results, explain unexpected results, audit trail, and so on. According to W3C semantic web, provenance is information about *entities* (datasets, physical, digital objects), *activities* (action/processes, generate new entities), and *agent* (take roles in an activity, persons, software) involved in producing a piece of data or thing.

Provenance describes the use and production of entities by performing some activities. The activities can be influenced by agents. Figure 2.4 illustrates the relationships of these core types by UML diagram, and we obtain it from W3C [161]. Provenance can be used to form assessments about its quality, reliability, or

trustworthiness. We discuss provenance (for scientific workflow) in detail in Section 4.2.

2.7.1 Prospective provenance

It models an abstract view of the workflow. We can consider it as a recipe for future data derivation. Prospective provenance in scientific workflows refers to capturing and documenting the expected steps and data transformations that will occur during the execution of a workflow. It involves specifying the inputs, outputs, and expected intermediate results of each step in the workflow as well as any relevant metadata, such as parameter values and software versions. Prospective provenance can help researchers ensure the reproducibility of their experiments and facilitate collaboration by providing a clear record of the workflow's design and intended behavior.

2.7.2 Retrospective provenance

Retrospective provenance in scientific workflows refers to the documentation of the history of the data and processing steps used in carrying out a scientific workflow after the workflow has been completed. It entails recording and storing information about the execution environment, the input data, parameters used, intermediate results, and output data generated by the workflow as well as the relationships between them. The retrospective provenance information can help to verify and audit the scientific methodology used in the workflow, to reproduce the results, to facilitate collaboration and sharing, and to enhance the scientific understanding of the research process.

2.8 Machine Learning Algorithms

Machine learning (ML) is used to perform a specific task with the help of computer systems without being explicitly programmed. It is a branch of Artificial intelligence (AI) that enables computers to mimic human behavior.

ML is the scientific study of algorithms and statistical models [120]. Machine learning algorithms are a set of mathematical models and techniques that allow computers to learn patterns and relationships in data without being explicitly programmed. These algorithms can be supervised, unsupervised, or semi-supervised, aiming to improve the accuracy of predictions, classification, and decision-making tasks.

Supervised learning algorithms use labeled data to train the model to predict outcomes for new data, while unsupervised learning algorithms identify hidden patterns or relationships in unlabeled data. Semi-supervised learning combines both, where a small set of labeled data is used in conjunction with a larger set of unlabeled data to train the model.

There are several popular machine learning algorithms, including linear regression, support vector machines, decision trees, random forests, neural networks, and deep learning. Each algorithm has its own strengths and weaknesses and is suited for different types of problems, data, and applications.

2.8.1 Neural Network

A neural network is a machine-learning algorithm modeled after the workings of the human brain. It consists of layers of connected nodes or neurons that process and transmit information. Each neuron is responsible for computing a weighted sum of the inputs it receives and passing the output through an activation function to produce an output. The structure of a neural network is typically composed of an input layer, one or more hidden layers, and an output layer. The input layer receives the raw data, such as images, text, or numerical data, and passes it to the hidden layers. The hidden layers perform computations on the input and pass the results to the output layer, which produces the final prediction or output.

Neural networks can be used in many applications like natural language processing, speech recognition, game playing, and decision-making. There are several types of neural networks, namely, *Feed-forward neural network, Convolutional neural network, Recurrent neural network* etc.

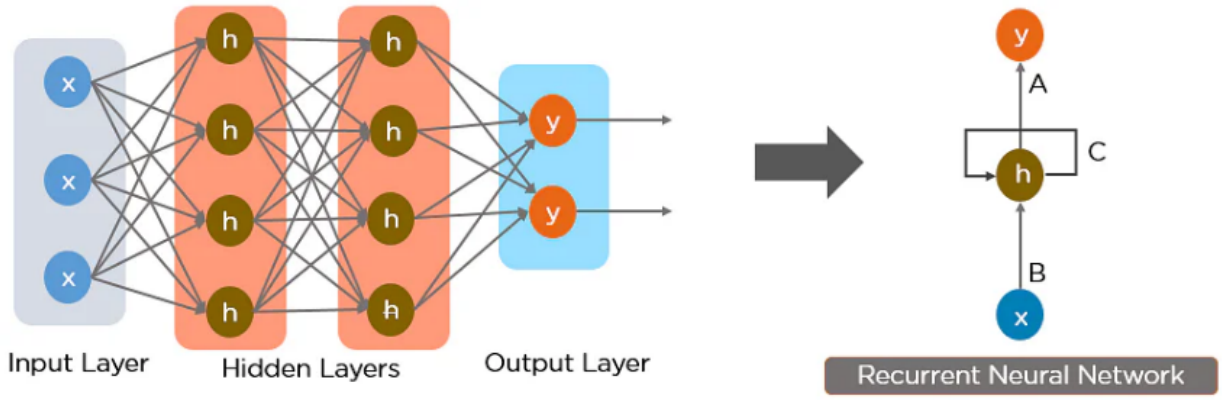

Figure 2.5: Simple recurrent neural network (Adapted from [17])

2.8.2 Sequential Models

Sequential models are a type of neural network architecture in which layers are stacked sequentially on top of each other linearly. Machine learning models that input or output data sequences are known as sequential models. Time-series data, text streams, and audio and video clips are some examples of sequential data.

The most common type of sequential model is the recurrent neural network (RNN), which uses loops in the network to allow information to be passed from one step of the sequence to the next. This makes RNNs particularly effective for tasks such as natural language processing, speech recognition, and time series forecasting. Another type of sequential model is the convolutional neural network (CNN), which is commonly used in image processing tasks where local patterns and features need to be detected.

In sequential models, the data depend on one another due to their sequential order. Sequential models are popular for natural language processing, time series prediction, speech recognition, voice recognition, and

automatic car parking. RNN and its variants, Autoencoders and Seq2Seq are some examples of sequential models

In a scientific workflow of Galaxy, the tools are connected sequentially. The sequential nature of these tool sequences encouraged us to apply similar techniques used for sequential data (text, speech, etc.) processing. Natural language processing, medical signals, clinical research, speech recognition, etc., apply deep learning techniques on sequential data in predicting future items obtaining good accuracy. Several studies [169, 111, 34, 20] show that recurrent neural network (RNN) provides a better result in sequential data processing.

Recurrent neural network (RNN)

RNN is a neural network that processes sequential data, such as time series or natural language data. An RNN has a memory component that allows it to capture dependencies and patterns across a sequence of inputs. Unlike feedforward neural networks, RNNs use loops in the network architecture to allow the output of the previous step to be fed back into the network as input for the next step. This feedback loop allows the network to maintain a memory of the earlier inputs and use that information to guide the prediction of the current output. Figure 2.5 shows a simple recurrent neural network, and we obtain it from [17].

The basic unit of an RNN is the recurrent cell, which contains a hidden state that is updated at each time step of the sequence. The hidden state is a vector that summarizes the information from the previous steps of the sequence, and it is used to generate the output at the current step. *Long short-term memory (LSTM) networks* and *gated recurrent units (GRUs)* are the variations of RNNs. GRUs work fine with longer sequences, and it has fewer gates than LSTM, so we mainly focus on GRUs.

Gated recurrent unit (GRU): GRU is a type of recurrent neural network designed to address the limitations of traditional RNNs, such as vanishing gradients and difficulties in capturing long-term dependencies. It can be considered as a variation of long short-term memory (LSTM).

Like RNNs, GRUs process sequential data by passing information from one-time step to the next through a hidden state. However, unlike standard RNNs, GRUs have a gating mechanism that controls the flow of information in and out of the hidden state.

GRUs consist of two gates: the reset gate and the update gate. The reset gate determines how much of the previous hidden state to forget, and the update gate decides how much of the new information to let in. These gates allow the network to remember or forget information from the previous time steps selectively.

GRUs can be used for a wide range of sequential data processing tasks, including language modeling, speech recognition, and image captioning. They are particularly effective in cases where long-term dependencies need to be captured, but the training data is limited.

Transformers: Transformers are a type of neural network architecture that aims to solve the problem of sequence transduction. They are based on a self-attention mechanism, which allows the network to weigh the importance of different parts of the input sequence when generating output. Unlike recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which process sequential input data by consuming one

element at a time in a fixed order, transformers are capable of processing the entire sequence in parallel, which makes them more computationally efficient.

The basic building block of a transformer is the self-attention mechanism, which allows the network to capture dependencies between different elements of the input sequence. The self-attention mechanism consists of three parts:

- **Query matrix:** applies a linear transformation to the input sequence to generate a set of query vectors
- **Key matrix:** applies a linear transformation to the input sequence to generate a set of key vectors
- **Value matrix:** applies a linear transformation to the input sequence to generate a set of value vectors

Each query vector is then used to calculate a set of attention weights, determining which elements of the input sequence are most important for generating the output. The output is then generated by a weighted combination of the value vectors, where the attention mechanism determines the weights. Transformers also use a positional encoding mechanism to help the network understand the ordering of elements in the input sequence, which is not explicitly captured by the self-attention mechanism.

The full transformer architecture consists of multiple layers of self-attention and feedforward neural networks, with residual connections and layer normalization between each layer to help with training. The output of the final layer is used to generate the final prediction or output.

The advantage of transformers is that they can learn long-term dependencies in the input sequences more effectively by allowing access to all of the information in the sequence rather than relying on fixed-size representations of the input.

3 Reusability Challenges of Scientific Workflows: A Case Study for Galaxy

Constructing a scientific workflow is a complex task for workflow designers. Scientists of diverse domains share their workflows by means of public workflow repositories so that other users can benefit from using them. Thus, scientists attempt to reuse existing workflows shared in workflow repositories. Unfortunately, several challenges prevent scientists from reusing those workflows. In this chapter, we thus first attempt to identify those reusability challenges. We also offer an action list and evidence-based guidelines to promote the reusability of scientific workflows. Our intensive manual investigation examines the reusability of existing workflows and exposes several challenges. Such challenges and our action list offer guidelines to future workflow composers to create better workflows with enhanced reusability. A significant percentage of workflows in the repository is mostly related to bioinformatics. So our study is mostly applicable to bioinformatics workflow, which may not be applicable to other domains such as chemistry, physics, astronomy, and earth science.

The rest of the chapter is organized as follows. In Section 3.2, we discuss about reusability from the view of our context. Section 3.3 discusses a few examples that motivate us to find the reusability challenges. We then present our study methodology in Section 3.4. Section 3.5 provides a detailed discussion of the study findings. We enlist our key findings and guidelines in Section 3.6. We discuss possible threats to the validity of our work in Section 3.7. Section 3.8 presents the related research works. Finally, Section 3.9 concludes the study with some future work direction.

3.1 Introduction

Scientific software capable of efficiently and effectively analyzing large-scale data has become requisite for many research programs exploring the world's most pressing and complex challenges, such as plant phenotyping and genotyping for food security, water quality prediction, and population-based cancer studies [93, 110]. Moreover, as the volume, variety, and velocity of data increases, extensive data experiments employing SWfMSs to explore data, plan experimental execution, and visualize the results are becoming more common [93]. In a workflow system, computational analysis of data encompasses multiple steps such as data preprocessing, quality control, quantification, and statistical analysis to transform raw data into scientific results. To accomplish this computational analysis and visualization, thousands of software tools have been

developed in SWfMSs. For example, only the Galaxy SWfMS has more than 8400 valid accessible software tools [144]. Constructing scientific workflows using this vast amount of tools has become a huge hurdle for scientists. Studies [93, 130, 126] suggest that despite decades of research, existing SWfMSs do not yet support the needs of the global scientific community to construct and adapt complex workflows quickly. While public datasets and pipelines proliferate, researchers remain unassisted in creating relevant analyses from these resources that remain largely underutilized [126]. It is difficult to understand what series of modules and connections need to be added to obtain the desired result without detailed knowledge of underlying computational components. Furthermore, working out which software tools and combinations are applicable or optimal in practice is often challenging. Thus researchers face difficulties in selecting practical and effective data analysis pipelines for a specific experimental design [139].

Analyzing scientific data is one of the most challenging issues scientists are facing today. As analyses get more complex and large, interdisciplinary groups need to work together, and knowledge sharing becomes essential to support effective data exploration [123]. So researchers developed several community frameworks for analyzing workflows. Also, scientific workflows often are complex, so sharing those employing public workflow repositories has become an essential issue for the community. As a result, public repositories are gaining importance as a means to share, find, and reuse such workflows [154]. Moreover, open science has emerged as a framework for improving the quality of scientific analysis. Transparent, accessible knowledge-sharing, and collaborative networks are essential components of open science. Scientific workflow communities are also tending toward open science and, as a result, they make many workflows available to the community in different repositories, i.e., Galaxy [75, 66, 70], WorkflowHub [45], myExperiment [80]. Other repositories [132, 167, 56] also have thousands of scientific workflows. Scientists share their workflows with these repositories so that other users can reuse them to solve their problems. Reusing workflows allow scientists to do new experiment faster, as the workflows capture valuable expertise from other scientists. Unfortunately, these workflows are not always reusable by other users. There are several [154, 106, 155, 104, 98, 38] works using available workflow repositories to reuse existing workflows in different ways, but there is a marked lack of research investigating (1) *the challenges of reusability of scientific workflow* and (2) *the approaches to overcome such challenges.*

Among SWfMSs, Galaxy [78], a web-based, integrated software framework, has gained momentum for analyzing large-scale datasets, and it is installed on over 169 research servers worldwide [96]. It is an open-source platform with data managing and reproducibility facilities for FAIR (*Findable, Accessible, Interoperable, and Reusable*) [164] data analysis. It enables researchers to use thousands of software tools from various domains and provides an interactive environment for researchers. Galaxy has a vibrant community to maintain servers, wrap tools, conduct workshops, implement needed features, and answer questions [67]. The number of users in the community is increasing rapidly. Community members can ask questions or provide answers to any questions in the community help forum [68]. Moreover, it is the most successful SWfMS [101] and intuitive to use. So, for conducting this research work, we mainly focus on Galaxy SWfMS, more specifically

Galaxy Main Server [72] workflows.

In this study, we follow three sequential steps for each sample workflow. First, we attempt to understand the purpose of the workflow by reading the name and annotation. Then we import the relevant workflow on our workflow list (you need to register to Galaxy for importing, the process is described in the replication package [7]) and try to reuse it. Finally, we record our findings on the challenges of reusing a workflow, and in case of failure to reuse, we investigate why it cannot be reused. We also record the actions necessary to make a workflow reusable. Our investigated results are mostly applicable to bioinformatics, and a few of them are related to statistics and visualization. Thus, our contributions are purely applicable to the bioinformatics domain. We answer three research questions and thus make three contributions to this study as follows:

- **RQ1: What are the challenges to reuse a scientific workflow? How can reusability challenges be measured?** We conduct an extensive manual analysis and investigate the reusability challenges of scientific workflows at the Galaxy Main Server workflow repository. We classify the reusability status of scientific workflows into *Reusable* and *Nonreusable*. We further sub-categorize the reusable workflows into *Reusable without modification, Easily reusable, Moderately difficult to reuse, and Difficult to reuse*.

- **RQ2: What percentage of scientific workflows can be reused or not with or without modifications?** We accomplish a detailed statistical analysis to determine what percentage of the workflows can be reused with or without any minor or major modifications to the corresponding workflows.

- **RQ3: How can we overcome the reusability challenges?** While we try to reuse scientific workflows, we face several challenges. To make the workflows reusable, we have to perform several minor or major changes. We record a set of actions.

3.2 Reusability

Reusability of a scientific workflow refers to using an existing workflow in some forms during the workflow design process. The necessity of reusability emerges in the scientific community due to the significant growth of tools, the complexity and heterogeneity of workflows, and sharing facilities in SWfMSs. There are thousands of tools in SWfMSs for performing several computational and visualization tasks, which are increasing rapidly. For example, consider the Galaxy SWfMS tools statistics shown in Figure 3.1. It is almost impossible for any workflow designer to remember these vast amounts of tool functionalities.

In addition, finding which tools and combinations are optimal in a workflow construction is often problematic. Scientists can reuse the shared workflows available at different repositories [75, 70, 66, 132, 167, 56] to mitigate these challenges. But reusing existing workflows has several challenges, which motivate us to do this research study.

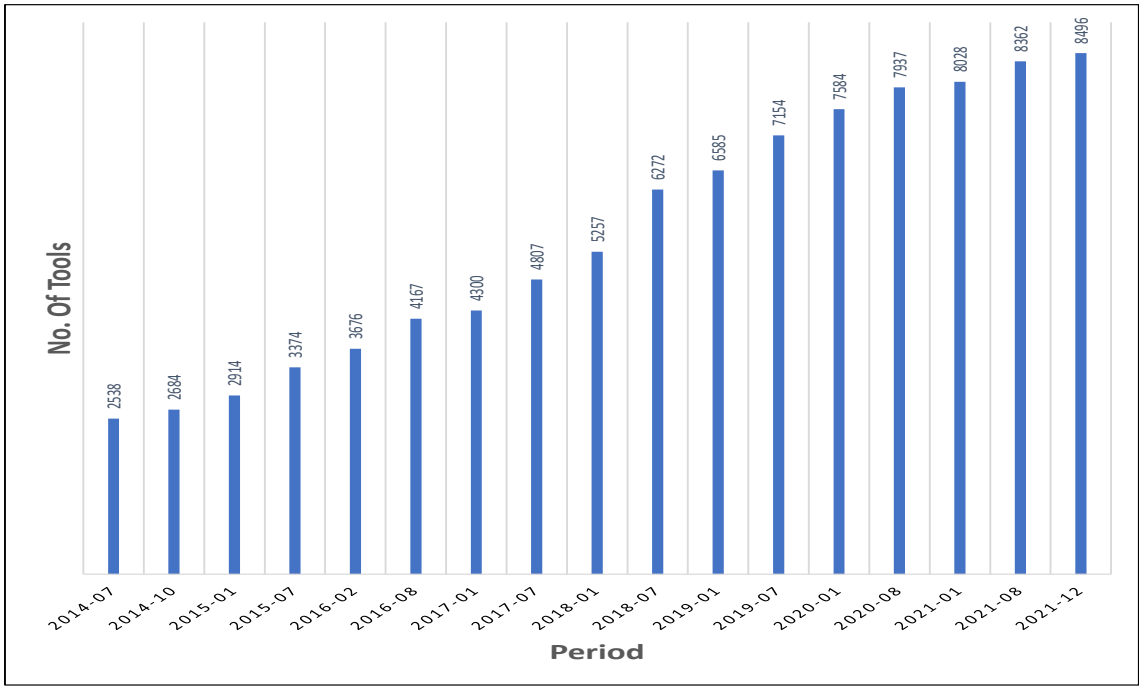

Figure 3.1: Galaxy SWfMS tools statistics over period

3.3 Motivating Examples

Scientific workflows published at scientific workflow repositories might not always be reusable. For example, let us consider a fragment of a scientific workflow shown in Figure 3.2. The workflow name is *RNA-seq analysis from SRA PE data (hg38, genecode.v24)*. Here, the workflow designer tried to do RNA sequence analysis using human genome SRA pair-end data. Any user can check the workflow using the Galaxy workflow repository [75] by searching the workflow using the name.

We attempt to reuse the workflow, but we face several challenges in doing so. Firstly, *genomespace-importer* tool support is unavailable in Galaxy as this tool is broken [143]. Then, in a workflow, the tools are connected and perform sequential operations, but there is no tool connection for this workflow. All the tools used are isolated. Moreover, one tool needs up-gradation, as indicated in Figure 3.2. The tool up-gradation can be done easily, but the other challenges are not resolvable.

Let us consider another example, as shown in Figure 3.3. The name of the workflow is *Creating a bed file of signficantly different genes*. Anyone can check the workflow using the process described in the previous example. In this workflow, the workflow designer performed at first a filter operation using a condition *(c14=='yes')* in column 14.

By exploring genome formats [160], we notice that there are three formats *(BED detail format has 14 columns, ENCODE gappedPeak: Gapped Peaks (or Regions) format has 15 columns, .2bit format has 16 columns)* which have 14 or more columns, and none of these formats contain value *yes* in column 14. Also, the workflow designer did not reveal anything about the input dataset. Another issue with this workflow is

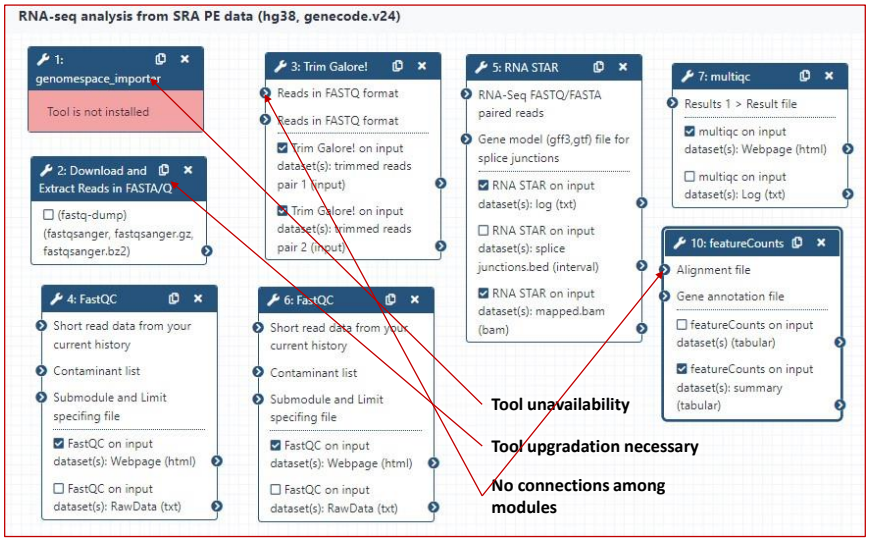

Figure 3.2: A segment of a scientific workflow (**Reusability status: Nonreusable**) (Adapted from [75])

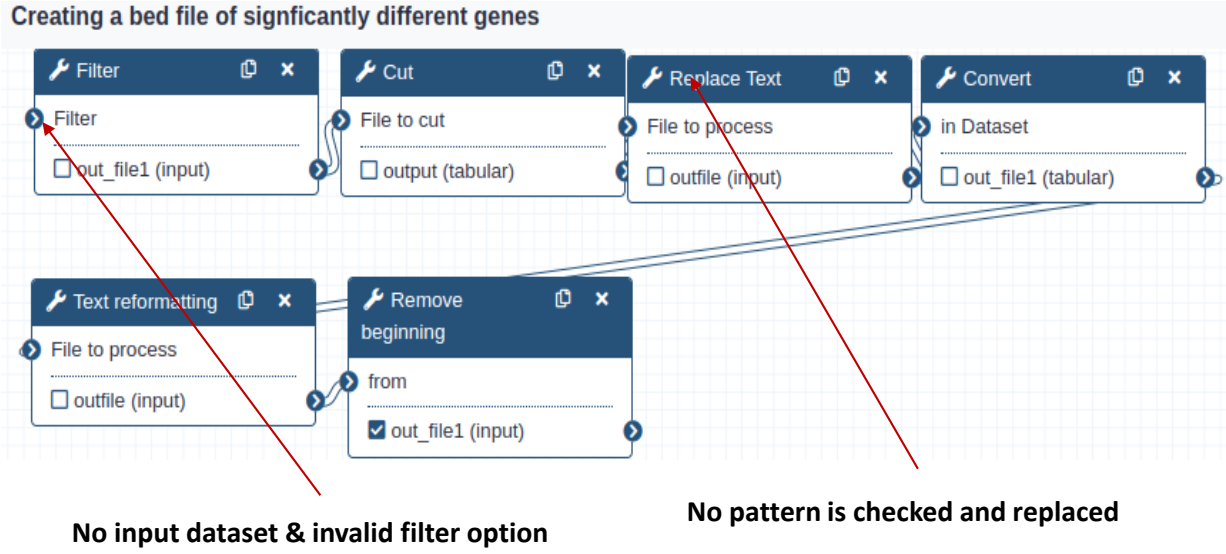

Figure 3.3: An example of a scientific workflow (**Reusability status: Nonreusable**) (Adapted from [75])

this workflow used a tool *Replace Text*, which is used to replace text in a specific column using a pattern. But here, the workflow designer did not provide any pattern to find and replace. These reasons make the workflow nonreusable.

Let us consider a reusable workflow example shown in Figure 3.4. The workflow name is *Exploring Iris dataset with statistics and scatterplots*. The workflow has the annotation (*Exploring the Iris dataset and visualizing features with two-dimensional scatterplots*) and tagging information (*iris, Groups, Scatterplot*).

Figure 3.4: An example of a scientific workflow (**Reusability status: Reusable**) (Adapted from [75])

At each step of action, a proper explanation is given for this workflow. Like where the data can be obtained and how to preprocess the data by converting format and removing the header, an intuitive data analysis process (how many species, count of species and differentiating the species). Finally, the owner visualized Iris dataset features with two-dimensional scatterplots. All activities are nicely explained in stepwise annotations. Anyone can explore this workflow and check the details from the Galaxy Main Server workflow repository [75] by searching using the name and annotation.

Easily reusable, moderately difficult to reuse and difficult to reuse workflow examples are demonstrated in Section 3.5.

3.4 Study Methodology

Figure 3.5 shows the schematic diagram of our conducted exploratory study. We, at first, select a statistically significant number of scientific workflows from Galaxy Main Server workflow repository [75] and then try to reuse them. We can reuse a percentage of workflows without facing any issues, but there are many workflows where we need to perform a list of actions to make them reusable. We also get a significant percentage of workflows that are nonreusable. The following sub-sections discuss different steps of our methodology.

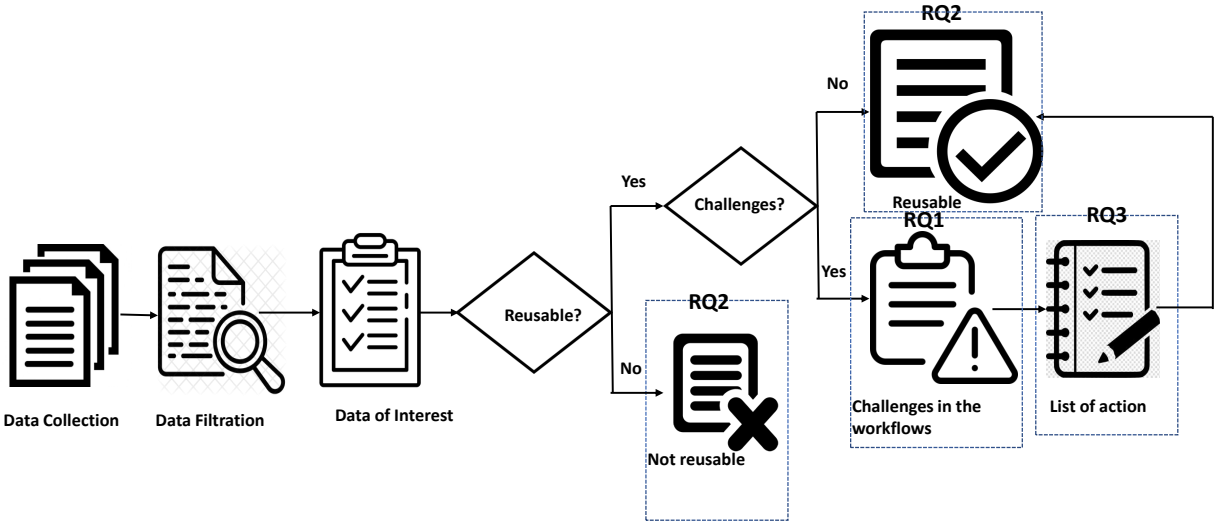

Figure 3.5: Schematic diagram of our exploratory study

3.4.1 Dataset preparation

Figure 3.6 depicts the data collection steps. We collected data from Galaxy Main Server Workflow repository [75]. This repository has 810 scientific workflows (June 2022).

At first, we discard such workflows that do not have a proper name or annotation. The workflow's name indicates the purpose of the workflow, and the annotation denotes the nature of activities occurring in them. By reading the name, any workflow designer should be able to understand what the workflow is doing. In addition, the annotation helps the workflow designer describe the workflow's particular functionality to the community so that they can easily understand the tasks to reuse. The annotation also helps to create a new workflow by assisting users in selecting suggestive components [76]. Our manual investigation finds 17.9% of workflows that do not have proper names/annotations.

A sample of improper names are *Unnamed workflow, Unnamed history, Test Workflow, Galaxy Test, Assignment, username, Workshop*, and in most of the cases for these workflows, the annotations are missing (only two workflows among all the improper naming workflows have annotations which are *test, 123*). They are also irrelevant. Naming/annotating is a qualitative measure, so we conduct an agreement analysis with our findings with another software researcher *(Ph.D. student and have a fundamental knowledge of*

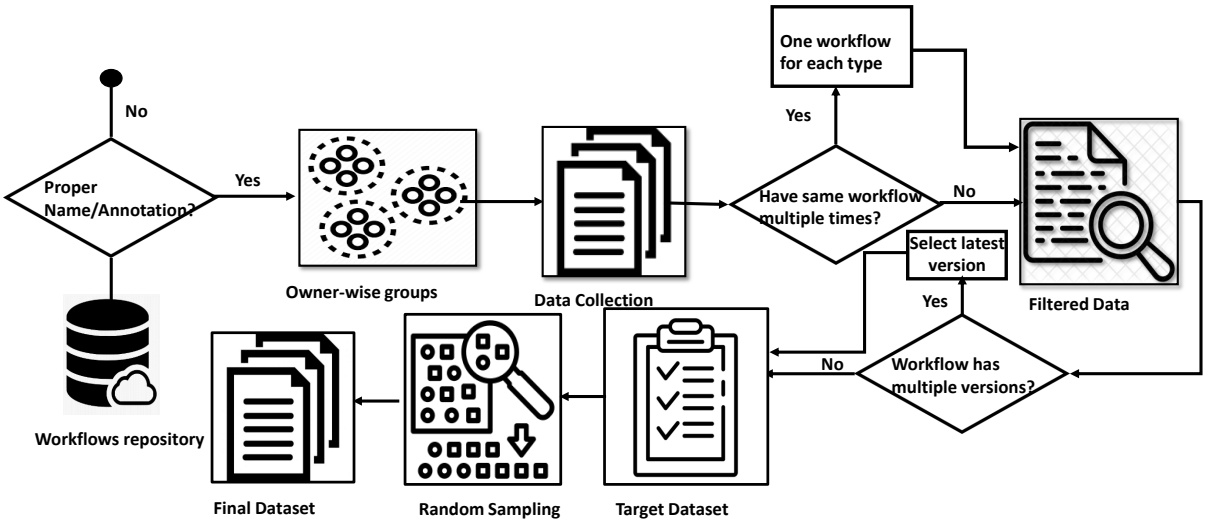

Figure 3.6: Selection of the dataset for study

scientific workflow and SWfMSs). We take $100 > 74$ (95% confidence level with 10% interval of 307 samples) workflows names/annotations information from the shared workflows name/annotation document, and we independently mark each name/annotation as proper or improper. We finally measure the agreement using Cohen's Kappa [37]. The value of κ is 0.86, meaning the agreement's strength is near perfect.

Then we make owner-wise groups so that we can consider most of the groups in our analysis which will mitigate biases. We find nearly 250 individual groups. We only select workflows from the groups where the number of published workflows is at least two. In this way, we discard 49 groups. The remaining groups act as a sample dataset of our study. If multiple groups have the same workflow multiple times, we pick one workflow and discard the others. Later, we check if a workflow has multiple versions (for each group); we take the updated one and discard the others. After discarding the above-mentioned workflows, finally, from the remaining 544 workflows, we randomly select $307 > 226$ (95% confidence level with 5% margin of error of 544 samples) workflows for our analysis and answer our research questions.

Replication: All experimental data and relevant materials are hosted online [7] for replication or third-party reuse.

3.4.2 Data analysis

We manually analyze 307 workflows. For each workflow, we attempt to understand the purpose of the workflow (the problem it is solving) and the usage of the tools. In addition, we check the parameter values of tools wherever relevant. Then we try to reuse every workflow. In many cases, we face several reusability challenges, so we need to perform trivial, minor, and major edits on the workflows to overcome the challenges. The Study Findings Section 3.5 describes all the things. A community forum can be a valuable tool to solicit user input and improve a product. It allows engaging directly with real users and understanding their needs and preferences, and finally addressing their concerns or suggestions. In order to identify the challenges

the users may face, along with reusability checking, we also check the community forum. The forum has thousands of issues from users using Galaxy. The challenges include tools not working, different results on the same workflow, complexities in providing tools parameters, missing functions, long-running queries, etc. The users also ask for help constructing workflow or using any particular tool in the forum. The community members responded to most of the questions and provided different solutions for each question. We consider the questions while identifying the challenges, and we check the answer to make the action list to resolve the challenges.

3.5 Study Findings

By exploring the 307 workflows, we find several reusability challenges, and to overcome these challenges, we perform several activities. Based on the effort level, we classify the reusability status. We ask three research questions in this study and answer them carefully with the help of our data analysis findings.

3.5.1 RQ1: What are the challenges to reuse a scientific workflow? How can reusability challenges be measured?

We divide **RQ1** into two parts **RQ1(a) and RQ1(b)** respectively and answer them separately. Table 3.1 enlists the challenges we face in reusing a workflow, and Figure 3.7 shows our measuring approaches to reusability challenges.

Answering RQ1(a): What are the challenges to reuse a scientific workflow? We attempt to reuse our sampled workflows. However, we face several challenges, and sometimes, we get many workflows that are nonreusable due to several non-trivial reasons. Table 3.1 shows the challenges we face in reusing a workflow. A workflow may experience multiple challenges. We are discussing each challenge briefly as follows.

Obsolete tools in the workflows: There are many obsolete tools in workflows. Tool supports are unavailable for these tools (i.e., Galaxy removed them from its tool repository). There are several reasons for tool removal. The tool may be replaced by a more robust alternative tool; sometimes, the tool may have several bugs, or the same activities can be performed alternatively. By checking each workflow manually, we identify that 12.38% of workflows have one or more deprecated tools. These obsolete tools hinder reusability.

Tools Update/Upgrade: Software tools update/upgrade is a very common thing, and workflows tools are not apart from it. Bug fixing, resolving security issues, adding new features, and performance improvement lead to the tools update/upgrade. By analyzing workflows, we find that 39.09% of workflows need tools update/upgrade to make the workflows perfectly reusable.

Workflows cannot be presented as a DAG: In a scientific workflow, the tools are connected sequentially, and the output of one tool can be the input of one/more tools. In most cases, the workflow should be presented as a directed acyclic graph. But we find 10.10% of workflows where the tools are not connected,

which eventually violates scientific workflows characteristics. Missing tools connection is a very common challenge for reusing a workflow.

Table 3.1: Reusability challenges of scientific workflows

Sl No.	Reusability Challenges
1	Obsolete tools in the workflows (12.38%)
2	Tools Update/Upgrade (39.09%)
3	Workflows cannot be presented as a DAG (10.10%)
4	Incomplete & Empty workflows (0.77%)
5	Workflows loading failure (3.91%)
6	Incompatible tools connection (2%)
7	Invalid operations in the workflows (1.54%)
8	Missing/Incorrect values for tool parameters (0.77%)
9	Multiple independent workflows in a workflow (3.58%)
10	No information about the input dataset (1.54%)
11	Misleading workflows (0.77%)

Workflows loading failure: We find 3.91% of workflows that returned an internal server error (failed to load a workflow in SWfMS). These workflows are nonreusable. Actually, we were unable to check the tools and connections they contained.

Incompatible tools connection: In workflows, the tools are connected sequentially, and the output of one tool is the input of others. So the data type of the output tool and associated input tools need to be compatible. But by analyzing, we find that 2% of workflows contained incompatible tool connections.

Multiple independent workflows in a workflow: A workflow can contain multiple sub-workflows, but we notice that some workflows have multiple independent workflows (in a workflow, there have multiple workflows) which are not connected at all. These workflows are doing different operations. We find 3.58% of such workflows in the observed sample. This also increases the challenges of reusability.

There are also several other challenges of reusability, i.e., there are workflows that are doing invalid operations (cutting a column that is not available in the dataset and filtering with irrelevant data, redundant, irrelevant operations). We also find some workflows where no information is available about the input dataset; also, it is not understandable by reading the name or annotation they provided. A few workflows do not perform the tasks as it is mentioned in the workflow name. We identify these workflows as misleading workflows. We also get several workflows where the parameter values are missing (Replace Tool is used, but no value is given in finding and replacing patterns) or incorrect values (Filtering is done by a value not compatible with the dataset column). We identify that 5.54% of workflows have these challenges by analyzing our sample workflows.

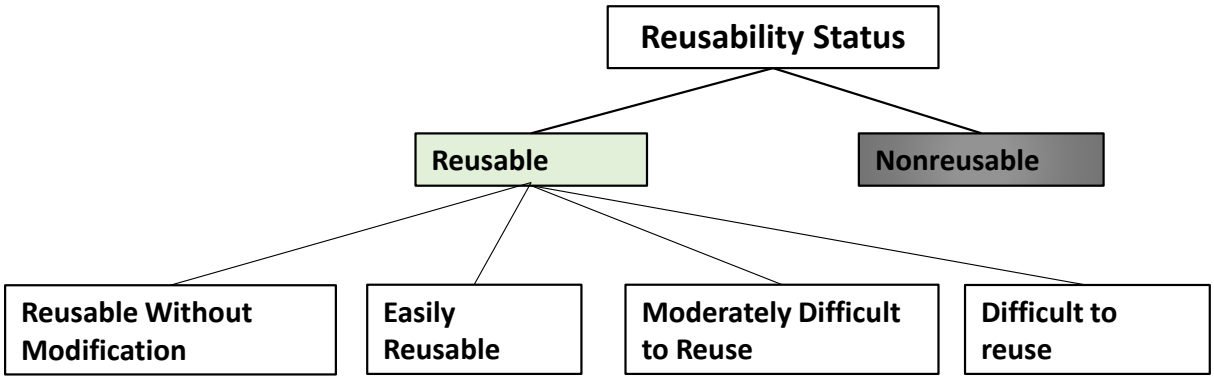

Figure 3.7: Classification of reusability status

Answering RQ1(b): How can reusability challenges be measured?: We divide the reusability status (of workflows) using two different dimensions. These are reusable and nonreusable. Based on the effort we need to make a workflow reusable, we further classify the reusable workflows into four sub-categories. Figure 3.7 shows our sub-classification of the reusable category.

In the following section, we discuss the reusability status in detail.

Reusable: A significant percentage of workflows can be reused without any modification. But there are many cases where workflows require some modifications. We, therefore, classify the reusable status into four more sub-categories based on the complexity of the modifications and the time required to reuse the workflow. Figure 3.7 shows our sub-classification of the reusable category.

- *Reusable without modification (RWM):* The given workflow is complete, stand-alone, and requires no explicit action to reuse. The workflow in Figure 3.4 shows such an example. Any user can use this workflow to distinguish the iris species through visualization.

- *Easily reusable (ER):* These workflows can be reused by performing some modifications that are comparatively less complex (*adding a few connections, importing known input data, tools update/upgrade*) and less time-consuming. The activities we need to perform here are to update/upgrade one or more tools, add a connection between tools, and sometimes change the label of the steps. We spend about 5-8 minutes reusing workflows from this sub-category. Figure 3.8 shows such an example. The name of the workflow is *QualStats-Boxplot-Distribution*. This workflow creates a quality statistics report for FASTQ data and shows the result in boxplot and distribution chat. We need to import the *FASTQ* dataset as input and connect it to the *Compute quality statistics* module.

- *Moderately difficult to reuse (MDR):* These workflows require much modification and are comparatively more complex (*Workflow has few errors, requires manual debugging to find issue-creating tools, finding out alternative tools for deprecated tools*) and time-consuming. For example, consider a segment of workflow shown in Figure 3.9. The workflow name is *Workflow constructed from history 'imported: ngs-wes-illumina-basic-genmed-2018.01.'*. While trying to reuse this workflow, we face errors, then we

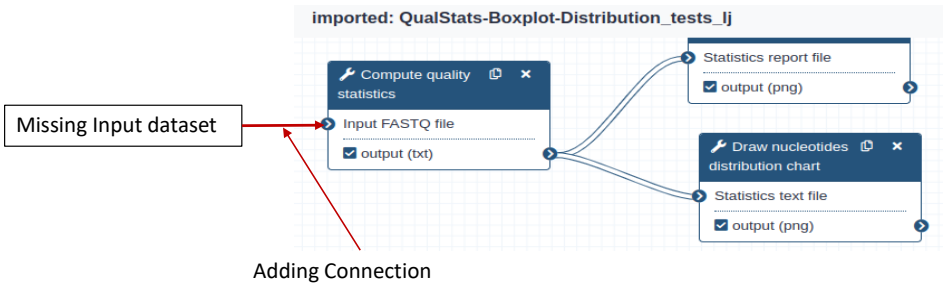

Figure 3.8: An example of a scientific workflow (**Reusability status: Easily reusable**) (Adapted from [75])

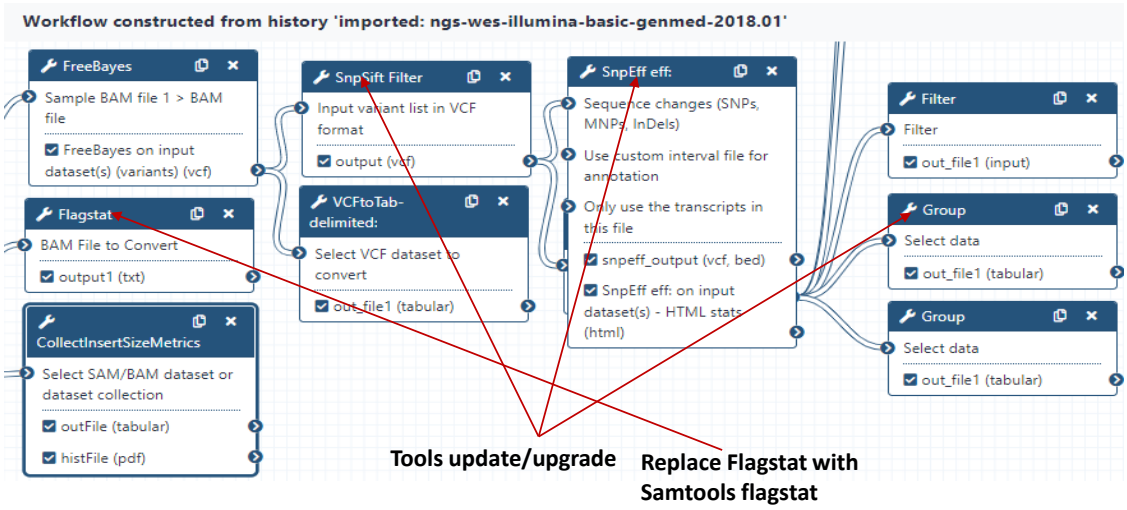

Figure 3.9: A segment of a scientific workflow (**Reusability status: Moderately difficult to reuse**) (Adapted from [75])

debug the workflow and find that the *flagstat* tool is the reason. We replace this tool with an alternative tool named *Samtools flagstat* by keeping the same outcome. We also need to update *SnpSift Filter, SnpEff eff, Group* tools. We spend about 15-30 minutes reusing each workflow from this sub-category.

- *Difficult to reuse (DR):* These workflows are complex and require a lot of modifications (*Workflow has several errors, lots of module connections, failure of tool up-gradation, lots of debugging necessary*), and are very time-consuming. Consider a segment of workflow shown in Figure 3.10. The workflow name is *workflow constructed from history 'SNP for Trio'*. We need to update/upgrade *VCFfilter, SnpEff eff, Group, Select first* tools. After that, we face issues with saving the workflow. Then we debug the workflow and find that *VCFfilter, SnpEff eff* tools up-gradation failed. Finally, we remove these tools from the workflow with connected modules and later import the latest versions of these tools and create

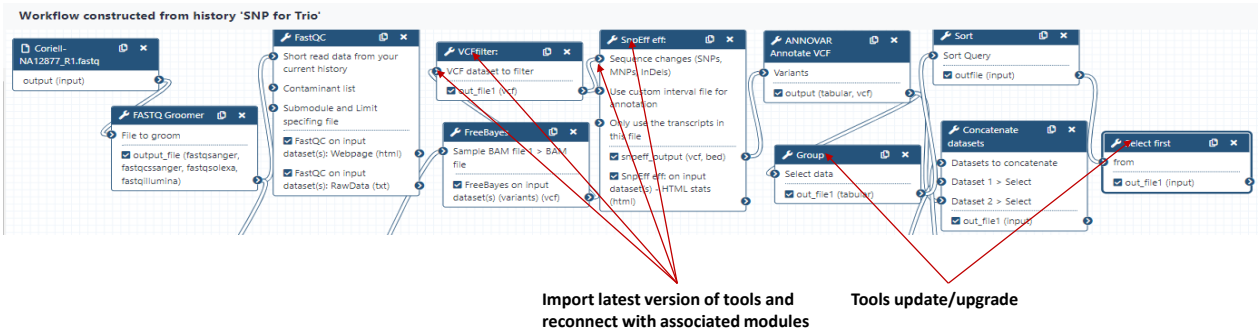

Figure 3.10: A segment of a scientific workflow (**Reusability status: Difficult to reuse**) (Adapted from [75])

the associated connections. These types of workflows took 25-45 minutes to resolve the issues.

Nonreusable (NR): We cannot reuse these workflows even after several minor and major modifications. We, two authors, take part in the manual analysis. Cross-validation is used to determine how well a system will generalize to new data. It is typically used in machine learning models. It may not be the best approach for cross-validating subjective opinions because opinions are usually based on personal experiences and beliefs rather than data. However, while it may be challenging to cross-validate the subjective opinions, it is important to seek ways to ensure their validity and reliability, especially in areas where opinions play a crucial role in decision-making. In our study, we found a significant number of workflows nonreusable. Hence, it is important whether our claim is right or wrong. Thus, at first, one author (with more than two years of research experience with scientific workflows & SWfMSs) marks the workflows which can not be reused as nonreusable. Then, another author (with more than seven years of research experience with scientific workflows & SWfMSs) randomly takes 22% of workflows from the list and tries to reuse them. In 90% cases, we get a similar result (and for the rest of the 10%, the disagreement occurred because another author considers that a part of these workflows can be reused). As we get similar results, we can say that our opinions are consistent and reliable. Our analyzed dataset is available in the replication package [7]. We show an example of a nonreusable workflow in the Figure 3.3.

3.5.2 RQ2: What percentage of scientific workflows can be reused or not with or without modification?

Figure 3.11 shows the statistics on the reusability status of the workflows and the modification effort we need to make workflows reusable.

Answering RQ2: According to our analysis, 225 (73.3%) workflows among 307 can be reused. On the contrary, 82 (26.7%) workflows are nonreusable due to the challenges discussed in **RQ1(a)**. Figure 3.11a shows the statistics on the reusability status of the workflows. The ratio of reusable status according to effort

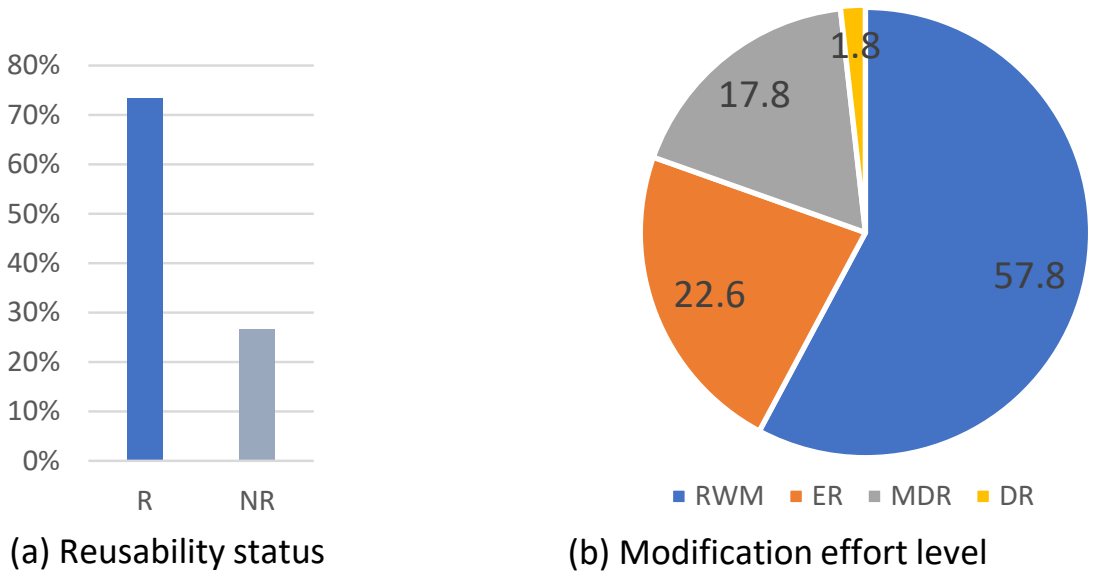

Figure 3.11: Reusability status (workflows) & modification effort level (Reusable workflows).

level is depicted in Figure 3.11b. Fortunately, 57.8% of workflows are reusable without any modification. 22.6% of workflows require minor edits and are easy to use. However, 17.8% of workflows need more changes compared to easily reusable workflows, and only 1.8% of workflows require significant modifications, which are tough to reuse.

3.5.3 RQ3: How can we overcome the reusability challenges?

Table 3.2 shows the action list we need to perform to make a workflow reusable.

Answering RQ3: A significant number of workflows can be made reusable by conducting some activities on the workflows. We perform several actions to make a workflow reusable through this exploration study. A workflow may need multiple actions based on the issues it contains. Table 3.2 shows the action list. We discuss each activity below.

Identifying proper input dataset: All operations in a scientific workflow are performed on scientific data. Proper input data selection for a workflow is a crucial task. We identify several workflows where information about the input dataset is missing. In addition, we find a bunch of workflows where input datasets' modules are also unavailable. We identify such datasets by understanding workflow activities. For example, consider a workflow from our sample. The workflow name is *mt analysis 0.01 strand-specific (fastq double)*. By searching the name, anyone can check the workflow using [75]. Here, there is no input dataset. But from the workflow, it is understandable that the input dataset is a *FASTQ* file. But in many cases, this is not understandable; then, the workflow becomes nonreusable. It is better to annotate the dataset and its source in the workflow step.

Table 3.2: Actions to make a workflow reusable

Sl No.	Action List
1	Identifying proper input dataset
2	Updating/Upgrading tools
3	Finding out alternative tools support for obsolete tools
4	Checking and modifying tools connection
5	Debugging to find the issue creating tools and connections
6	Changing step labels
7	Removing unnecessary portions of workflows

Updating/Upgrading tools: Unfortunately, many workflows are missing the updated/upgraded versions of tools. Workflow publishers should check for any updates/upgrades of the tools used in the workflow.

Finding out alternative tools support for obsolete tools: There are many workflows where tools become obsolete in our sample dataset. If a workflow contains an outdated tool, we try to find another alternative tool to make the workflow reusable. For example, a tool *flagstat* becomes outdated, and an enhanced version *Samtools flagstat* is available to support the task of *flagstat*. So we replace *flagstat* with *Samtools flagstat*.

Checking and modifying tools connections: The modules are connected in a workflow, but we notice quite a few workflows where modules are not connected. Moreover, some connections get broken due to tool up-gradation also. Therefore, we add connections among tools wherever necessary.

Debugging to find the issue-creating tools and connections: We identify several workflows where we need to perform stepwise debugging to find out the issue-creating tools and figure out the issues behind this. Finally, we modify the workflow to make it reusable.

Changing step labels: We find a few workflows where several input datasets have the same step labelings for a single workflow. As a result, it generates errors while executing the workflow. Renaming the step labels resolves this issue.

Removing unnecessary portions of workflows: SWfMSs support sub-workflows, but we identify some workflows where each workflow contains multiple independent workflows. Keeping the necessary one, we discard the other workflows. Our replication package [7] includes several samples of this kind of workflow.

3.6 Key Findings and Guidelines

Constructing a scientific workflow is a complex and time-consuming task. Reusing can also be very difficult if it is not organized appropriately. Throughout this study, we identify several challenges of reusing a workflow, and below, we provide some key insights to make a workflow reusable without facing any difficulties.

- **Proper description of input dataset and expected outcome:** We identify a large number of

workflows where we do not get a clear description of the input dataset. It would be better if the workflow designer could provide a brief annotation about the input dataset, like the sources of data, the nature of the data, and the usage. It is good to have a clear description of the workflow's output.

- **Step-wise annotations:** We find several workflows where the number of modules (tool connections) is too high. Sometimes it is difficult to understand the activities of each module. If step-wise annotations of each module could be provided, it would be easier for others to understand the operations of the workflow and which eventually helps reusability.

- **Update/Upgrade workflow tools on a certain interval:** SWfMSs are evolving extensively. Therefore, tool updates/upgrades become a common phenomenon for SWfMSs to enhance tool capacities, improve performance, and resolve bugs and security flaws. So it would be better if the workflow designers could check their shared workflows on a certain interval to see if any tools require an update/upgrade.

- **Proper parameter values for tools:** Selecting appropriate parameter values is essential to obtain the correct outcome from a workflow. It is recommended to use appropriate parameter values and mention the reasons for choosing them.

- **Check before publishing:** We find workflows in our sample where tools are not connected, multiple irrelevant workflows in a workflow, empty workflows, the purposes mentioned and activities are different, many unnecessary operations, and so on. So before publishing a workflow in the repository, it is recommended to check all the mentioned reasons.

3.7 Threats to Validity

Validity of research is concerned with the question of how the conclusions might be wrong, i.e., the relationship between conclusions and reality [125]. It refers to how sound the research design and methods are. Threats to external validity relate to the generalizability of findings to other populations and settings. We manually analyze a limited number of scientific workflows from the Galaxy Main Server workflow repository, so our results may not generalize to all other workflows from different repositories. Nonetheless, replication of our study with other workflow repositories may prove fruitful. However, we believe that our findings of reusability challenges and actions can be generalized to other workflows from different repositories. But we recommend readers not to over-generalize our findings.

Threats to *internal validity* are influences that can affect the independent variables with respect to causality [165]. The workflow's reusability status (reusable, nonreusable) is threatened by the subjectivity of our classification approach. Thus, we cross-validate our obtained results when a workflow was nonreusable and agreed with 90% similarity.

Threats to *construct validity* are related to the design of the study [40]. We take a sample of 307 workflows; thus, the research outcome may be changed by different samples. To mitigate this issue, we accomplish owner-

wise grouping and select workflows randomly from each group. But we avoid taking the same workflows from multiple groups so that we can explore workflows from various perspectives.

In our research, we do not perform any actual workflow user study, which may cause some biases in our obtained results. Without a survey, researchers may miss important data that could influence the study results. Thus, our results may not be fully applicable to the practitioner's users.

Another potential threat to our study is that our analyzed workflows dataset mostly contain bioinformatics-related workflow. Thus, we suggest not generalizing our results with other domains.

3.8 Related Work

Several research works have been done to design SWfMSs, provenance data management, querying and optimized storing of provenance, and developing querying language. But, the scientific community of workflow management still lacks the reusability of workflows effectively. To simplify creating error-free and effective workflows for scientific data analysis, the reusability of existing workflows can play a significant role. A few approaches have been taken for reusing existing workflows. Kumar et al. [106] developed a model using a deep learning approach by analyzing workflows available in the European Galaxy server [71] for suggesting tools while constructing a workflow. But they did not consider the issues the workflows may contain. For example, the workflows could be erroneous; there could have incompatible tool connections, obsolete tools, and many other challenges. Koop et al. [104] proposed techniques to help scientists construct workflow by automatically suggesting completion based on a database of previously created workflows for the Vistrails system. They developed a tool named *VisComplete*, but they only considered visualization pipelines. They also did not think about the challenges of reusing existing workflows. Junaid et al. [98] proposed a provenance recording and suggestion system for the quick and easy creation of error-free workflows. Their approach stored detailed provenance information in the provenance store and provided suggestions based on it to reduce user interaction and automate the design process. But they did not consider the execution of existing workflows. So, while creating workflows, their system can give unexpected tools suggestions. Zhang et al. [171] proposed an approach *Recommend-As-You-Go* to recommend services in a workflow composition process based on previous services usage history. They developed a network named *People-Service-Workflow* (PSW) that models workflows and their past usage relationships into a social network. But we investigate that the previously stored workflows have several issues to be reused without any modifications. Wings [79] can automatically track constraints and rule out invalid designs focusing on the core aspects of the experiments. It generates workflow candidates by searching and validating choices of datasets, parameter values, and software components. It creates multiple variations of workflow using different tools. But, this system ignored the presence of higher-order relationships [128] in tool sequences. Understanding a workflow is difficult due to its complex nature. To facilitate understandability and, therefore, effective reuse of workflows, Garijo et al. [76] identified workflow abstractions from several SWfMSs workflows. But their work did not

find the challenges of reusability and actions to overcome them. There is a marked lack of analyzing scientific workflow repositories' reusability challenges. To the best of our knowledge, ours is the first work investigating the reusability challenges and actions to overcome these challenges.

3.9 Conclusion and Future Work

In this study, we manually analyze 307 randomly selected scientific workflows from Galaxy Main Server Workflow Repository and investigate their reusability challenges. Explaining with evidence, we answer three research questions in this work. We classify reusability status into two major categories, namely reusable and nonreusable. We find that 73.29% of workflows can be reused, and 26.71% are nonreusable. We need to perform several minor or major modifications on 42.22% of reusable workflows, and 57.78% require no change. We also record the challenges and describe actions to overcome them. In the future, we plan to work with other available workflow repositories, identify their challenges, and resolve them. We have a plan to do a user study to understand the real scenario of workflow reusability. Then we will develop a workbench for workflows using reusable workflows to reduce the problems scientists face now.

4 Challenges of Provenance in Scientific Workflow Management Systems

Almost every SWfMS capture provenance data while a workflow or a part of it becomes executed. Scientists often need to run computational modules multiple times to investigate a particular dataset. In such frequently executed workflows, processing large datasets that include computationally expensive modules requires a long execution time. A significant inefficiency of current workflow implementation is re-reading/re-executing the same data multiple times. Proper management of provenance information allows scientists to reuse intermediate data and assist workflow construction. If the intermediate data can be reused, then it can speed up the workflow execution times and minimize storage size. Thus, in this chapter, we document the challenges of provenance management and reuse in e-science, focusing primarily on scientific workflow approaches by exploring different SWfMSs and provenance management systems. We also investigate the ways to overcome the challenges.

The rest of the chapter is organized as follows. In Section 4.2, we discuss about provenance in SWfMSs. Section 4.3 discusses the related work and our study in the context of related work. We then present our study design in Section 4.4. Next, Section 4.5 provides a detailed discussion of the study findings. Finally, Section 4.6 concludes the study with some future work direction.

4.1 Introduction

Scientific workflows (shortened as workflows in this work) and SWfMSs provide potential opportunities to solve complex, multidisciplinary, and large-scale computational experiments like Cybershake [85], Montage [15], Epigenomics [16], Coronavirus sequencing [140], cancer studies [168], molecular dynamics [151], and so on. However, many of these workflows have significant computational, storage, and communication demands and thus must execute on a wide range of large-scale platforms [44], from large clouds to upcoming exascale HPC platforms. In SWfMSs, scientists often need to rerun workflows for finer-grained analyses and improve the quality of scientific analyses, leading to the FAIR [164] (*Findable, Accessible, Interoperable, and Reusable*) datasets and analysis pipelines. In such frequently executed workflows, processing large datasets that include computationally expensive modules requires a long execution time. The more data to process, the longer the workflow may take, which may be days depending on the problem and HPC environment [54]. Due to the popularity of workflows, hundreds of workflow management systems have been developed, but proper

management (storing and reusing) of provenance information is still not up to the mark. Furthermore, given the increasing number of high-quality public datasets and pipelines, this lack of apparent compatibility threatens the findability and reusability of these resources [126].

SWfMSs are becoming popular due to their capacity to manage complex and diverse applications, and they have paved the way for scientists to accelerate many scientific discoveries. However, if the same workflow or sub-workflow needs to be executed several times, it will generate a high volume of undesired redundant provenance data. Due to the complex heterogeneous nature of data and large volume, a single experiment may demand a week to run, even in high-performance computing environments [39]. Handling such exponentially increasing data volumes and utilizing the technical resources for storage and computing are thus demanded in exploiting data-intensive computing in various application fields. Gathering provenance data from distributed workflows raises new and different challenges [42]. Also, a lack of proper coordination and broadly usable standard components lead to the ad hoc and isolated solution rather than adopting and extending existing solutions, which causes wastage of time and resources [3]. Many existing SWfMSs capture detailed provenance information. Each SWfMS has its particular approach for executing a workflow and capturing its provenance information. This provenance information generally consists of data and process dependencies introduced during a workflow run. It is crucial for enabling scientists to more easily understand, reproduce, and verify scientific results [47]. Detailed provenance information allows scientists to audit trails and verify and reproduce their results.

Research [106, 126, 152, 104, 98] has been done to reuse existing workflows in different ways. Also, there is a lot of research [31, 6, 122, 108, 102, 87, 86, 9, 138, 142, 27] about storing, querying provenance information, audit trails using provenance also reproducibility issues of a scientific workflow using provenance data. Workflow provenance assures the reliability and integrity of workflows and the data as they are routed in complex workflows. A proper provenance record is essential in many scientific experiments as it enables experiments to be systematically repeated and validated by others. The amount and cost of provenance information can be inversely proportional to the granularity; they can grow to be larger than the data it describes. Although many SWfMS systems have been developed, there is still a marked lack of research investigating (1) *the reusability of provenance information in a scientific workflow execution* and (2) *optimized storing (non-redundant) of detailed provenance information.* In current SWfMSs, if the same workflow/sub-workflow needs to be executed multiple times, the workflow cannot automatically use the previously stored results. It costs a massive amount of storage and execution time and makes the system inefficient to use.

Open science has emerged as a framework for improving the quality of scientific analysis. Transparent, accessible knowledge-sharing, and collaborative networks are essential components of open science. Scientific workflow communities are also tending toward open science and, as a result, made many workflows and datasets available to the community in different repositories [75, 167, 70, 66, 69, 132, 56, 74, 65]. In addition, some of the repositories [74, 69, 65] also share some provenance information. The future of scientific advancements mostly depends on the ability of scientists to comprehend the vast amount of data currently

being produced and acquired. Unfortunately, while public datasets and pipelines proliferate, researchers remain unassisted in creating relevant analyses from these resources that remain largely underutilized [126]. Although a vast amount of information is available to the community, due to several challenges, they are mostly nonreusable, especially for a workflow re-execution purposes. Managing data with provenance in a SWfMS to support the reusability of workflows, modules, and data is complex. Handling such components is even more burdensome for frequently assembled and executed complex workflows for investigating large datasets with different technologies (i.e., various learning algorithms or models). Provenance information can be reused to recommend intermediate states [32]. Properly managing provenance data in a SWfMS for efficient storing and facilitating workflow executions is crucial [32].

In this study, we investigate the challenges of provenance data and record the actions to overcome them. In particular, we present techniques for reducing provenance overload and making provenance data more reusable and fine-grained. Furthermore, we identify that if provenance data can be appropriately managed, it will help tremendously to mitigate resources and time usage, eventually reducing cost and making the data analysis process more effective and efficient. Finally, we try to answer three research questions and, in this way, make three contributions to this study as follows:

- **RQ1: What are the provenance challenges in SWfMS?** We check the provenance capturing mechanisms of several SWfMSs and investigate the challenges, especially regarding reusability and storing approaches.
- **RQ2: How can we overcome the provenance challenges?** We identify several challenges and recommend several actions to overcome them. We suggest optimized storing, avoiding redundancies, enabling data sharing, and so on.
- **RQ3: Can we use provenance data in real-time?** While executing a workflow, we propose to use existing provenance data instead of executing the workflow module wherever possible. It will save a workflow execution time, especially for long-running queries, and ensure reusability.

4.2 Provenance

The provenance is sometimes referred to as audit trail, lineage, and pedigree and contains information about the process and datasets used to derive the data product. Provenance provides essential documentation to preserve data, determine its quality and authorship, reproduce as well as interpret and validate the associated scientific results [43]. Provenance helps provide context to a given set of data by providing details such as who created or modified it, what source systems were used to generate it, and how it was processed over time. Some researchers like Bose al. [19], Buneman et al. [23] describe provenance in terms of primary data, while others, Sahoo et al. [146] and Srivastava et al. [153] describe it as metadata. Some other authors, like Holland et al. [91], argue that provenance differs from other forms of metadata as it is based on relationships

among objects. Cruz et al. [43] considered provenance data as the (semi-) or automatically and systematically captured and recorded information that helps users or computing systems to determine the derivation history of a data product, starting on its original sources and ending at a given repository.

Provenance storage is used to register provenance information. It can be stored locally or distributed. Centralized provenance storage keeps data in one single repository at one single location. They are easy to use, maintain, manage, and control for security purposes, but the single point of failure is the problem. In distributed provenance storage, multiple interrelated repositories are distributed over a computer network. The storage can be RDBMS or file system. Distributed provenance storage can be classified as homogeneous or heterogeneous based on the types of storage systems. The life cycle of provenance is shown in Figure 4.1, which we obtain from Belhajjame et al. [12]

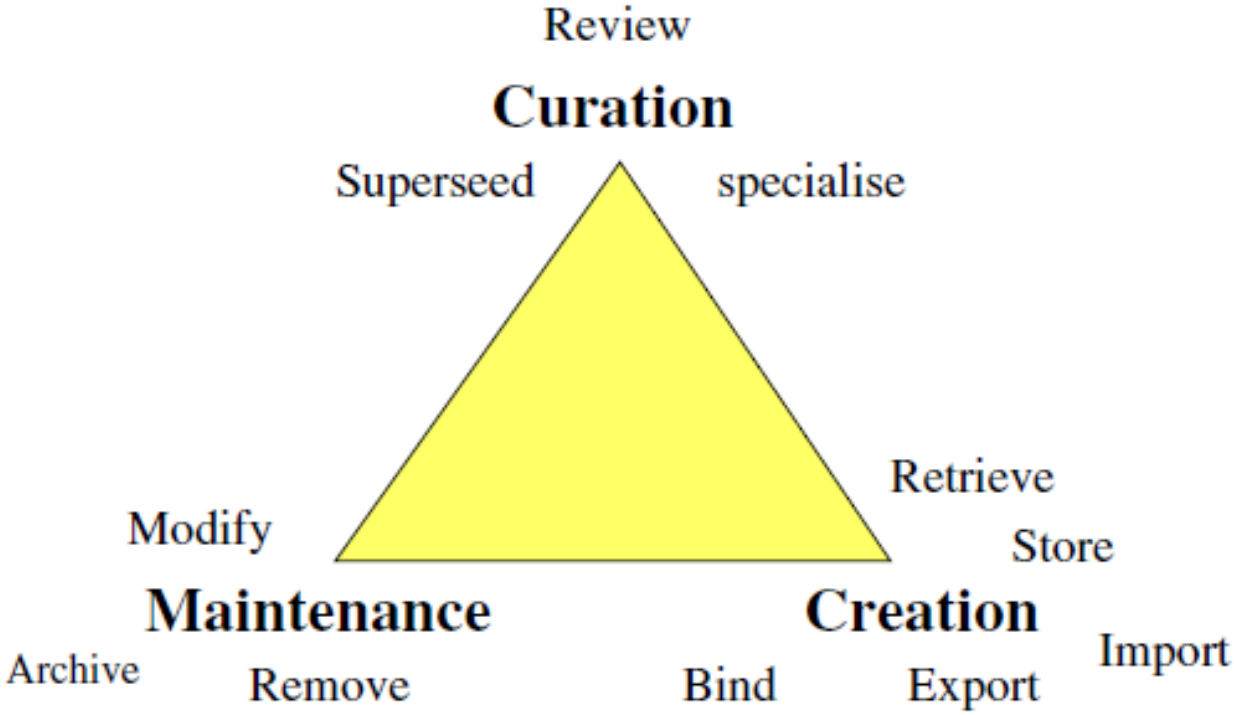

Figure 4.1: The life cycle of provenance (Adapted from [12])

An essential component of provenance is information about causality, and another key element is user-defined information which is not captured automatically but records important decisions and notes. Lazy and eager [22] approaches can be used to trace provenance data. Most SWfMSs used eager methods because provenance can be entirely determined here though it adds additional overhead to compute the output and to store annotations in the repository. There are three key components in a provenance management solution: first, the capturing mechanism, then the data model for representing the provenance information, and finally, the infrastructure for storing, accessing, and querying provenance. Two distinct forms of provenance are *prospective* (captures a computational task's specification) and *retrospective* (captures the steps executed as

well as information about the environment used to derive specific data product).

Prospective provenance is the practice of recording and tracking information about the data created or used by a workflow before it is actually executed. Retrospective provenance, on the other hand, tracks and records data after a workflow has been executed in order to provide an audit trail for any changes made to datasets within a workflow. In short, prospective provenance helps capture data at its source, while retrospective provenance captures data after it's already been processed.

In provenance, the dependency relationships among data products and the process that generate them are essential. In SWfMSs, provenance-gathering mechanisms are either attached or integrated. They are responsible for storing provenance data using a particular provenance model.

Although researchers tried to build a standard open provenance model for capturing the provenance data, provenance is tightly coupled to SWfMSs. Thus, scientific workflows' provenance concepts, representation, and mechanisms differ massively among SWfMSs. SWfMSs capture complex analysis processes at various levels of detail and systematically collect provenance information for the derived data products so they can be queried later.

4.3 Related Work

This section discusses related work on SWfMSs provenance based on reusability, storage, and querying systems. At the end of this section, we discuss our study in the context of related work.

4.3.1 Storing, querying, and reusing workflow provenance

Our study is a step towards helping scientists diminish their hassle in managing storage and enabling reusability by skipping executing earlier executed modules. For most SWfMSs, provenance management has become a vital functionality. There are several approaches to managing the provenance of scientific workflows. We are discussing some of them below.

RISP technique [31] focused on re-using intermediate stage results generated by previously executed workflows. This technique optimizes storing of workflow data using the mining association rule [2]. A data management scheme to facilitate workflow assembling and executions with different parameter configurations with a GUI is proposed in [32]. Anand et al. [6] focused on easy-to-use and efficient approaches for accessing and querying provenance information. A high-level query language (QLP) is tailored for expressing provenance queries. They provided formal semantics for the language and presented novel techniques, Immediate Lineage Edges (I), Immediate Edges and Dependency Closure (IC), and Immediate Edges and Closure via Pointers (ICP) for efficiently evaluating queries.

Kepler system [5] designed a provenance recording system (*Provenance Recorder (PR)*) to avoid complex configuration processes or the scientist implementing a complex API. To give more flexibility, they made their collection facility parametric and customizable. They allowed the user to choose between various levels

of detail and even save all of the provenance data needed to recreate a workflow result when the workflow is used as a part of the scientific discovery. Debugging a workflow is also possible in the implementation phase of workflow development. Marinho et al. [122] proposed *ProvManager*, a provenance management system for scientific workflows. ProvManager leverages provenance management at the experiment level by integrating different workflow executions from multiple workflow management systems. It eases the gathering, storing, and analyzing of provenance information in a distributed and heterogeneous environment scenario. As part of the UK's myGrid project, a workflow workbench Taverna [173, 172] is developed. Taverna's provenance model captures locally generated provenance data, and external provenance information gathered from third-party data providers. One advantage of this system is it supports overlaying secondary provenance over the primary logs and lineages. Taverna focused on the semantic web of provenance and showed how it could be mined by a provenance usage component, Provenance Query and Answer (ProQA).

Lim et al. [108] designed a provenance model that models both prospective and retrospective provenance as an extension to the Open Provenance Model (*OPM*). They implemented a relational provenance store to store, reason, and query prospective and retrospective provenance, which is captured via the proposed provenance collection framework. Provenance trails in the Wings/Pegasus system [102] focused on creating and executing large-scale scientific workflows, which involved lots of computations over distributed shared resources. Their approach uses semantic representations to (1) *describe complex scientific applications in a data-independent manner*, (2) *automatically generate workflows of computations for given data sets*, and (3) *map the workflows to available computing resources for efficient execution.* Here, workflow instantiation provenance can be queried using SPARQL [141], and workflow execution provenance can be queried using SQL. The PReServ/PASOA system [87, 86] supports the recording of interaction provenance, actor provenance, and input provenance with the provenance recording protocol. Provenance capturing through operating systems using Berkeley DB and XML database is focused on PASS [92] and Earth System Science Server (ES3) [62].

Barga et al. [9] introduced a layered model to represent workflow provenance that allows navigation from an abstract model of the experiment to instance data collected during a specific experiment run. They developed a method called *REDUX* to explore the benefits and challenges of automatically capturing experiment provenance, along with methods to store the resulting provenance data efficiently. REDUX uses the *Windows Workflow Foundation* (WinWF) as a workflow engine. The VisTrails system [28] represented an initial attempt to improve the scientific discovery process using XML and relational database technologies for provenance management. VisTrails addressed the visualization problem from a data management perspective and managed the data and metadata of a visualization product. It can support scientists in navigating through the space of workflows and parameter settings for an exploration process.

The provenance models proposed by the semantic web community for data-driven workflows capture retrospective provenance but underspecify the workflow structure. An underspecified workflow structure may misinterpret scientific experiments and preclude the workflow's conformance checking, eventually restricting

provenance. To overcome these challenges [25] proposed a formal lightweight and general purpose specification model for the control-flows involved workflows and integrated it with both ProvOne [41] and PROV-DM [13] provenance models.

Sometimes, large-scale experiments may demand a week to run, even in high-performance computing environments. So it becomes unviable to analyze provenance data only after the end of the execution. However, scientists can use run time provenance to monitor workflow execution and take action before execution end. For example, Costa et al. [39] worked with representing and sharing runtime provenance data to improve experiment management and analyze scientific data generated by parallel workflow execution in different environments adopting cartridges. Their works support runtime analysis, evaluate the status of each parallel task, take actions to improve workflow reliability and performance, spare financial resources, and steer the execution status at any time.

Apart from mentioned works CombeChem [63, 158], Mindswap [82, 83] and VIEW [33, 109] systems worked with provenance management. Swift [174], and Chimera [60] systems introduce a Virtual Data System (*VDS*) to use provenance for tracking the data derivation history, on-demand data generation and re-generation, and data product validation. Chimera [60] combines a virtual data catalog for representing data derivation procedures and derived data with a virtual data language interpreter that translates user requests into data definition and query operations on the database. They coupled the Chimera system with distributed data grid services to enable the on-demand execution of computation schedules constructed from database queries.

4.3.2 Our study in the context of related work

Several available SWfMSs use general-purpose relational RDF, structured files, relational tables, OWL schemas, or virtual collections to manage and query provenance. For this study, we plan to use a relational database to store provenance data and optimized SQL to query the provenance information. We mainly focus on two things.

1. elimination of redundancies based on the pattern and database schema semantics. This approach can be used in any general-purposed RDF stores or relational tables.

2. we also plan to store data, modules, and invocations information at a granular level to re-use it to ensure the re-usability of provenance data.

Our target is to provide a provenance management approach that eases the gathering, storage, and analysis of provenance information so that scientists can use provenance data without putting the burden on adaptations.

4.4 Study Design

This study aims to investigate the extent to which SWfMSs and different provenance management systems work mechanisms and discover the challenges of managing provenance data. We first explore state-of-the-art SWfMSs' provenance capturing, storing, and usability studies for this study. Then, we explore several provenance management models and the necessity of exposing provenance in open science (we review a representative set of widely used systems). Finally, we demonstrate the challenges and suggest possible actions to overcome them.

For practicing open science, Galaxy SWfMSs enabled users to share workflows with provenance data into several workflow repositories [74, 69, 65]. Galaxy has an internal proprietary provenance model and captures both prospective and retrospective provenance information; also, there are facilities for adding manual annotations (important decisions and notes, usually used to understand the meaning of data products or scientific applications). But Galaxy allows data redundancies, and the facilities for reusing the provenance data for workflow re-execution are unavailable. For example, if one user needs to execute a workflow multiple times, the current Galaxy system executes it from the beginning. Therefore, it cannot reuse the provenance information for re-execution. Details analysis is provided in the study finding section. We also go through the Galaxy repositories [74, 69, 65] and identify most of the workflows, as well as datasets, are redundant, which hinder SWfMSs' performance and cost enormous resources.

VisTrails' [28] provenance-management system provides infrastructure for data exploration and visualization. Using a relational database management system VisTrails uses an action-based provenance model to capture changes to parameter values and pipeline definitions. It supports backward and forward chains of reasoning. The VisTrails interface allows scientists to query, interact and visualize the process history. Though it supports the flexible reuse of workflow pipelines, collaborative exploration, and a flexible annotation framework, it does not support the reusability of provenance data for workflow re-execution. In VisTrails, the change-based provenance model records information about modifications to a task akin to a database transaction log. It also stores redundant data and allows redundant operations.

In Taverna, we notice two classes of provenance, one describes workflow-related entities like services, workflows, and sub-workflows, and the other describes workflow executions. It used a graph model to capture provenance information. In Taverna, the processors can exchange data by reference, which allows, among other things, to reduce the amount of data the workflow enactor has to convey between services, thereby supporting data-intensive processes. Taverna stores provenance metadata using RDF/XML stores, and here, multiple annotations can coexist and be associated with the same resource.

In Kepler [4], the specification of a workflow instance must be saved to the provenance model every time the workflow is executed, along with runtime information, which incurs high storage overheads and negatively impacts query performance. Karma provenance framework [150] stores both prospective and retrospective provenance using XML and relational databases. It supports dynamic workflows and explicitly models data

products' derivation history. It provides a light-weight and scalable implementation to meet the core needs of recording and querying for these provenance graphs over hundreds of thousands of service invocations and data products but allows data redundancies.

Several approaches have been developed for capturing and modeling provenance. For example, some models used traditional file systems, and others used relational database tables. In the file systems, the advantage is that users do not need additional infrastructure to store provenance information. The relational database provides centralized, efficient storage that a group of users can share. But there are several issues with storing, accessing, and querying provenance data.

4.5 Study Findings

Although provenance models differ in several ways, including their use of structures and storage strategies, they all share an essential type of information: processes and data dependencies. Provenance is relevant to a wide range of domains and applications, so it is crucial to identify the problem of systematically capturing and managing provenance for computational tasks. Without provenance, it is nearly impossible to reproduce and share results, validate results with a different set of input data, understand the operation process, and solve a complex problem collaboratively. Therefore, we conduct this study and gather the following findings in this study.

4.5.1 RQ1: What are the provenance challenges in SWfMSs?

By exploring state-of-the-art techniques of provenance management, we gather a set of challenges in managing provenance data. We enlist the challenges in Table 4.1

Answering RQ1: By doing this exploratory study, we identify the following challenges.

Data redundancies : Data redundancy means multiple copies of the same information spread over multiple locations in the same database. Data redundancies inflate the size of the database and create data inconsistencies. If data is redundant, then data maintenance becomes tedious and problematic. By exploring several SWfMSs and provenance models, we find that they all support redundant data storing in provenance.

Table 4.1: Provenance challenges in SWfMSs

Sl No.	Challenge List
1	Data redundencies
2	Lack of reusability
3	Data querying mechanisms
4	Provenance data sharing
5	Provenance overload

Lack of reusability: Data reusability lessens the response time, and reusable data allows first-mover

advantages. Due to the open science policy, many scientists make their data available to the community. Still, if a workflow/workflow module needs multiple re-execution, current provenance management systems do not reuse the previous data and make new execution which causes several problems like storage issues, data fetching and updating, and many more issues.

Data querying mechanisms: The ability to query provenance efficiently helps knowledge reusability, which helps compare and understand differences between different tasks. Current provenance management systems mostly use XML, RDF, and relational databases for storing and querying data. Unfortunately, the data querying mechanism is not user-friendly in most SWfMSs. For querying provenance data, most SWfMSs use SQL, Prolog, and SPARQL, which can be awkward and complex to write.

Provenance data sharing: Scientific studies often require a heterogeneous set of data, and these data sets are collected by independent research over many years, which are not accompanied by rich enough semantic information. We notice several workflow repositories where provenance information is shared, but they are difficult to interpret as there has no annotation or usage purpose, or explanation.

Provenance overload: Sometimes, a workflow execution can take multiple steps, and a module may need multiple executions. Current SWfMSs store all executed data. In this case, the information stored for a single workflow can be extensive. Thus, provenance overload can be a problem for these systems.

4.5.2 RQ2: How can we overcome the provenance challenges?

We enlist a set of actions to overcome the provenance challenges. Table reftab:provchallenaction contains the approaches to overcome the provenance challenges scientist usually face in using SWfMSs.

Answering RQ2: In SWfMSs, scientists need to work with a very high volume of data, and therefore, infrastructure for effectively and efficiently querying provenance data is an essential component of a provenance management system. In order to resolve the challenges scientists usually face for provenance, we identify the following measures:

Provenance database normalization: Structuring a relational database using normal forms reduces data redundancy and improves data integrity. Structuring provenance information into multiple layers enables normalized representation, which avoids storing redundant data. Current SWfMSs provenance systems contain vast amounts of redundant data. To get rid of it, we are suggesting using 3NF [35], or BCNF [36].

Enable data reusability: Current SWfMSs and their provenance management systems use provenance information primarily for reproducibility and audit-trail purposes. But data reusability can reduce the resource usage and time to perform an operation because it will avoid processing of same data multiple times.

Effective data sharing: Proper access to data is essential for an efficient, progressive, and self-correcting scientific ecosystem. Though open science encourages more open access to and use of data as it helps collaboration among teams and communities, there is still a significant lack of shared data in workflow communities. Also, we find redundant and erroneous shared data in several workflow repositories. Therefore,

Table 4.2: Overcoming provenance challenges

Sl No.	Action List
1	Provenance database normalization
2	Enable data reusability
3	Effective data sharing

we are encouraging adequate data sharing in the community.

Along with the mentioned reasons, we also encourage fine-grained data storing, an intuitive and interactive interface for provenance queries, a generic provenance model so that data can be shared among SWfMSs, and the usage of the optimized query to analyze provenance data.

4.5.3 RQ3: Can we use provenance data in real-time?

Using provenance data in real-time can reduce provenance management cost-effectively. In the following, we discuss the real-time usage of provenance data.

Answering RQ3: One of the primary perspectives of our study is to ensure provenance data, more specifically, retrospective provenance data reusability in scientific workflow execution. The FAIR principle also encouraged more open access to and use of data to solve problems, and many researchers considered reusability the primary concern. If we can store provenance data using data normalization and share the data following FAIR guidelines, then we can ensure data reusability.

4.6 Conclusion and Future Work

Data-centric computing is increasingly becoming essential for scientific analysis, and the FAIR principles have represented a meaningful way forward for open science datasets. Efficient data storing and re-usability are becoming inevitable components in SWfMS as workflow may become very large or the same workflow needs to be executed several times. In this study, we identify provenance challenges; more specifically, we focus on optimized storing and reusing provenance data. We also describe ways to mitigate these challenges. In the future, we will build a provenance data storing system that will be compatible with any SWfMS and ensure optimized storing and reusing of provenance data along with reproduction of results from the earlier executions, explaining unexpected results and efficient data sharing.

5 Recommending Tool(s) for Scientific Workflow Management Systems

Workflow management has become essential in scientific research as it facilitates the automation of complex operational processes. However, selecting the right tool(s) can be overwhelming due to the diverse range of options available. It is essential to use the proper tool for each workflow step to achieve desired results. To make this possible, we propose a tool(s) recommendation system to recommend tool(s) using a machine learning-based approach (gated recurrent units neural network) to help scientists create optimal, error-free, and efficient workflows by analyzing existing workflows (most of them are related to bioinformatics) in various workflow repositories. We perform a preliminary evaluation by successfully composing three scientific workflows using our tool(s) recommendation system in Galaxy as a proof of concept. The precision of our model is 98%.

5.1 Introduction

Workflows are becoming essential in analyzing scientific data. There are hundreds of SWfMSs, such as Galaxy, Pegasus, VizSciFlow, SciWorCS, VisTrails, Kepler, or Askalon, where researchers can create, collaborate and share workflows for their analysis. They can create it by themselves or can use existing shared workflows from different available repositories. Figure 5.1 shows an exemplary scientific workflow from the myExperiment repository, which is constructed in Taverna. This workflow extracts the plain text content of PDF files supplied to the input port. Scientific workflows make data analysis simple and convenient and have several advantages.

However, several challenges make workflow construction difficult; scientists focused on reusing existing workflows. They tried to find the relevant workflows by applying several state-of-the-art techniques, and they could get a set of workflows from the repositories. But, collecting the appropriate workflow from the collection of workflows is also difficult.

Furthermore, we have identified several issues in reusing these workflows. Then the necessity of a recommendation system in the scientific workflow comes. The objective of recommender systems is to help people find suitable, exciting, and newly released products. The task of recommender systems is to turn data on users and their preferences into predictions of users' possible future likes and interests [117]. Recommender systems have been used in online shopping, travel booking, and media service providers for many years. It

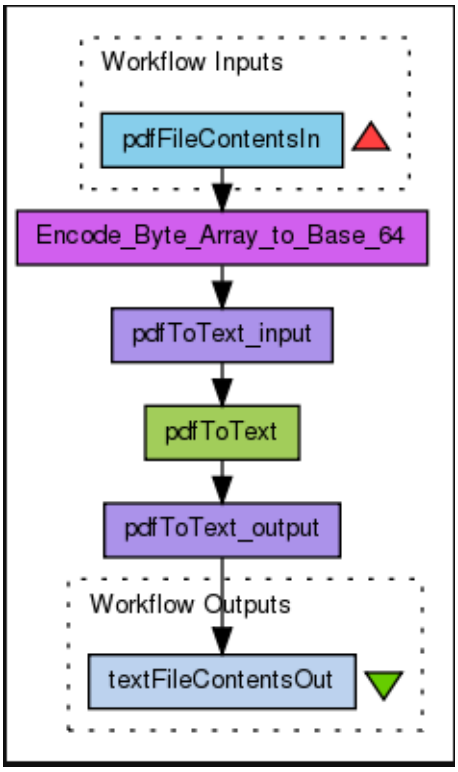

Figure 5.1: Sample scientific workflow. (Adapted from myExperiment [132])

is also used extensively in the scientific community for scientific literature searches to help scientists explore relevant and recent papers quickly. The recommended products are primarily chosen based on past usage and purchasing patterns. Decreasing the data storage and processing cost led to the significant improvement of recommender systems by making predictions on the available data. Commercial companies like Alibaba and Amazon use their customers' preferences to select products from an extensive collection. Production companies like YouTube and Netflix suggest new songs, movies, etc., based on the previous usage of the customer. In short, recommender systems make life smoother for the users to find appropriate things among the ocean of products. Recommender systems differ in how they analyze data sources to develop notions of affinity between users and items, which can be used to identify well-matched pairs [127]. The successful usage of recommendation systems by companies in diverse areas to find relevant things encouraged us to create a tool(s) recommendation system in SWfMSs.

Current workflow repositories support keyword queries matched against workflow annotation, tags, and titles given to the workflow while uploading by the workflow designers. The search quality critically depends on the quality of the described annotations and other parameters associated with the workflows. Another way of searching workflow is the definition of a workflow itself. The structure or topology of the workflow, together with the attributes defined on the workflow's modules, is used by structure-based methods of workflow comparison. The advantage of structure-based search is they do not require any additional information apart from the workflow itself. Users usually identify workflows that roughly match their needs using keyword

search; then, they use structure-based search. But this way has several issues which led me to do this research.

A critical issue for a workflow is that it is not always state-of-the-art or even valid at all. A workflow may run on a system without creating an issue and produce some results, but the generated result may not always be correct due to the wrong selection of tools and techniques. So to resolve this problem, a system is essential that can recommend useful tool(s) for constructing pipeline(s) in a workflow. There are several [75, 70, 66, 132, 167, 56] publicly available workflow repositories, and there are also several tool(s)/workflow(s) recommendation systems [106, 104, 98] available which usually recommends based on the previous usages of the tools. But tools and techniques are becoming updated every now and then. Also, many new optimized tools are being added to the workflow management system, many tools are becoming deprecated, and many workflows have errors. So recommendation system based on these workflows will surely recommend the wrong tool(s) at some point in the workflow construction. So in this study, we develop a tool(s) recommendation system. While developing the system, we consider the usage frequencies of tools, high-quality and low-quality sequences, usage periods, workflow naming, annotations, and workflow correctness.

SWfMSs contain thousands of tools to analyze scientific datasets, and they are increasing rapidly. We also notice that scientists are now more eager to publish their workflows and datasets so that other users can benefit from them. Galaxy SWfMS has several workflow histories repositories [74, 69, 65, 75, 70, 66], and Galaxy is the most popular SWfMS to date. Thus, for conducting this study, we explore Galaxy-published workflows and identify that a significant percentage of workflows have issues with them, and we are expecting similar results for the other SWfMSs. Hence, recommendation systems based on the existing workflows may provide undesired suggestions. For our case, in the beginning, we identify the poorly designed workflow using manual (inconsistent but running workflows) and automated (tools compatibility, obsolete tools, broken workflow) processes and discard them so that our system can suggest the best combinations of tools while constructing a workflow.

At first, we explore all available Galaxy workflow repositories to gather the reusable workflows and discard error-containing, irrelevant and obsolete workflows. After that, using gated recurrent units (GRU) neural network, we build an intuitive tool(s) recommendation system using reusable workflows. Most of the workflows in Galaxy repositories are mostly related to bioinformatics. Thus, our system works well in the bioinformatics domain. But we expect our tool(s) recommendation system will work with other domains too. We also have a plan to work with other SWfMSs in the future. We believe that our work will contribute in the following ways:

- Our analysis data will help raise awareness to create a workflow appropriately so that researchers can understand and reuse it without facing too many difficulties.

- Our system will save time researchers waste in creating erroneous or less optimal workflows by choosing improper tool(s) which produce unexpected results.

- It will relieve researchers from memorizing a vast amount of tools and increase the accessibility of the tools.

- It will promote high-quality tools by checking the previous (a certain period) usage frequencies and downgrading those having lower usage frequencies.

5.2 Recommendation System

Generally speaking, a recommendation system is an information filtering technique. It provides users with information that they may be interested in. In software engineering, it is a software application that provides information items estimated to be valuable for a software engineering task in a given context [77]. Recommendation systems make the system development process more efficient and less expensive, lower developers' cognitive load, and help them make better decisions, eventually improving the system's quality. Recommendation systems are emerging to assist software developers, from reusing code to writing effective bug reports and in various other activities. Figure 5.2 shows an example of the working principle of a conventional recommender system which we obtain from [115].

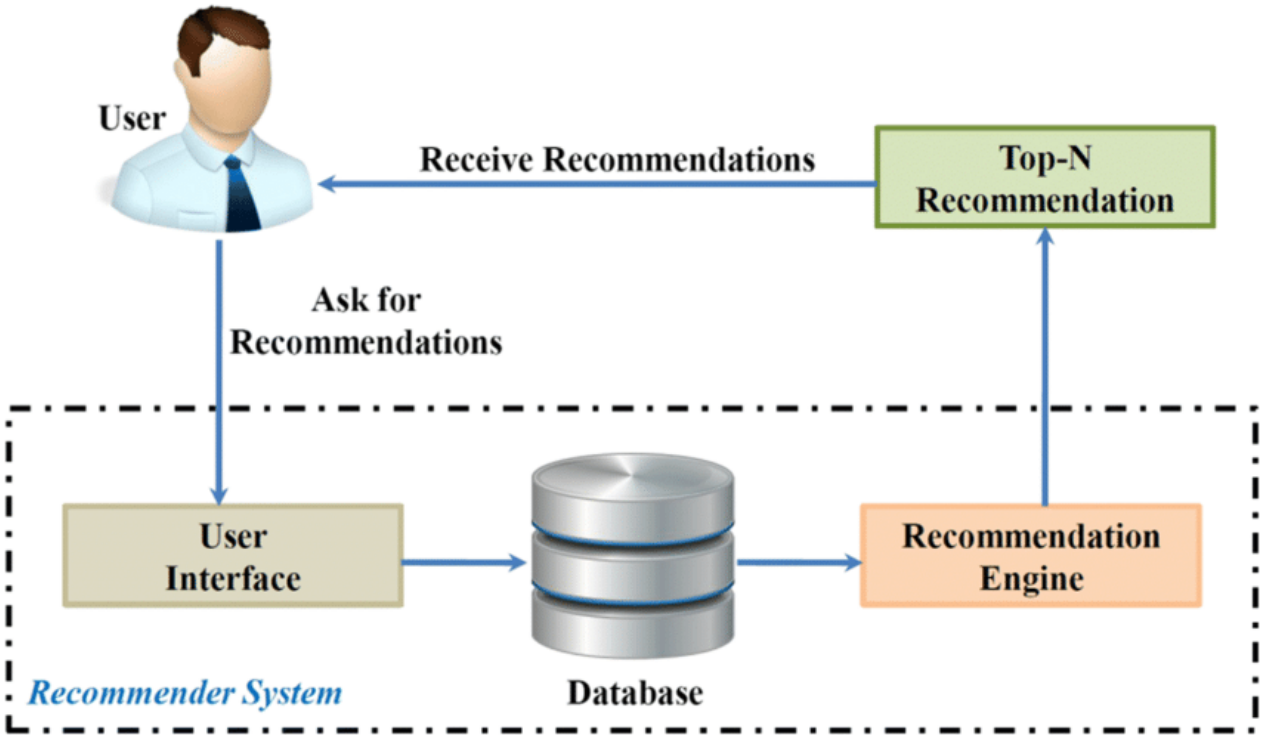

Figure 5.2: Traditional recommendation system (Adapted from [115])

Various factors are involved in making a decision in the recommendation system. The major approaches of recommendation systems are collaborative filtering [57], content-based filtering [116], and knowledge-based recommendation [24]. Typically, a recommendation system depends upon its filtering algorithms, datasets,

methods, taxonomies, etc. Figure 5.3 shows the workflow of a recommendation system which we obtain from [121].

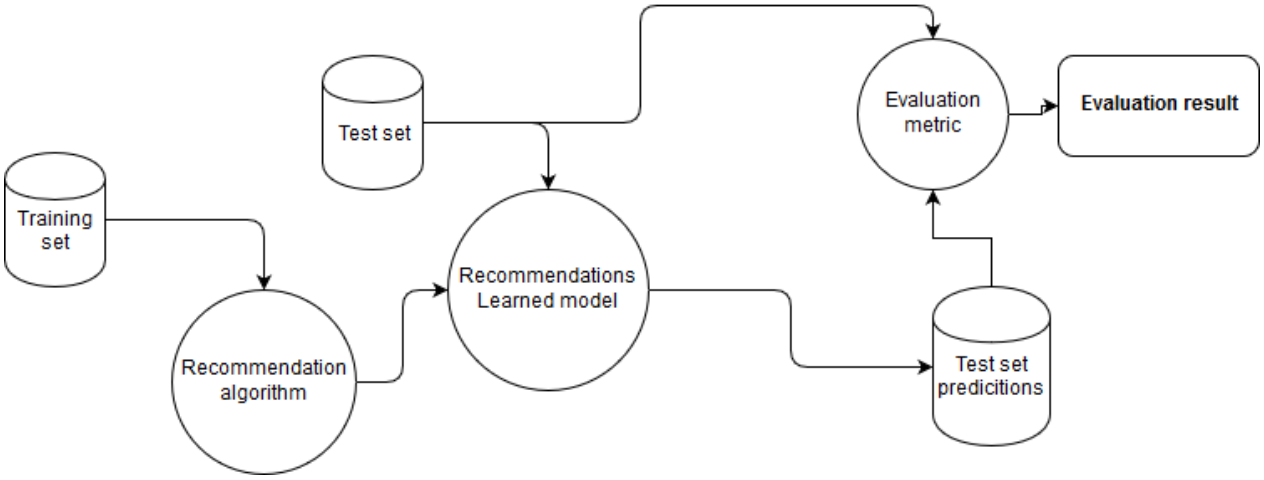

Figure 5.3: Recommendation system workflow (Adapted from [121])

In the paradigm of digital services, users suffer due to information overload problems, and recommendation systems become a decision support tool to address this problem. Large companies like Netflix, Amazon, YouTube, and Google Play, and social and professional networking sites like Facebook and LinkedIn sit on a vast volume of collective information of their users, and recommendation system is an inevitable part of all these companies. The exponential growth in the number of published articles also necessitates the importance of a recommendation system to help researchers to explore relevant and recent papers easily. A recommendation system can resolve the information overload problem by retrieving information using similar users' preferences and interests. The successful usages of recommendation systems in software engineering, scientific literature search, e-commerce, social media, and everywhere encourage us to create a tool(s) recommendation system for scientific workflow.

5.3 Sequential Learning on Workflows

Sequential learning is an approach to scientific workflow in which the tasks are completed in a certain order which is designed by workflow developers. Here, each task is depended on the outcome of the previous one or more tasks. Typically, a scientific workflow can be represented as a directed acyclic graph. The tools in a workflow are connected sequentially. In sequential learning, the sequences are a data structure where the data points are dependent on each other. Tasks like google translate, machine translation, self-parking, and question answering are all sequence modeling tasks. A sequence modeling problem is to predict the next word. The sequential nature of workflows tools connections encourages us to apply similar learning techniques used for other sequential data, such as natural language and speech recognition. Many studies have shown that variants of RNN provide better accuracy in sequential learning. Therefore, in this study, we

use gated recurrent units, a variant of RNN, to create the tool(s) recommendation system. RNNs can deal with variable-length sequences, maintain sequence order, share parameters across the sequence, and keep track of long-term dependencies.

5.4 Motivational Example

Scientific workflow construction is a difficult task. Scientists need prior knowledge about available tools for performing their desired activities. Gathering knowledge about massive amounts of tools and finding the most appropriate combinations of tools for a particular workflow takes a lot of effort from a workflow developer. For example, consider a segment of COVID-19: variation analysis reporting workflow shown in Figure 5.4. The name of the workflow is *COVID-19: variation analysis reporting*. The input of the workflow is VCF datasets of variants produced by the variant-calling workflows [95] and generates tabular reports of variants by samples and by variant. The output also contains an overview plot of variants and their allele frequencies across all samples. Anyone can check the details of the workflow in [166].

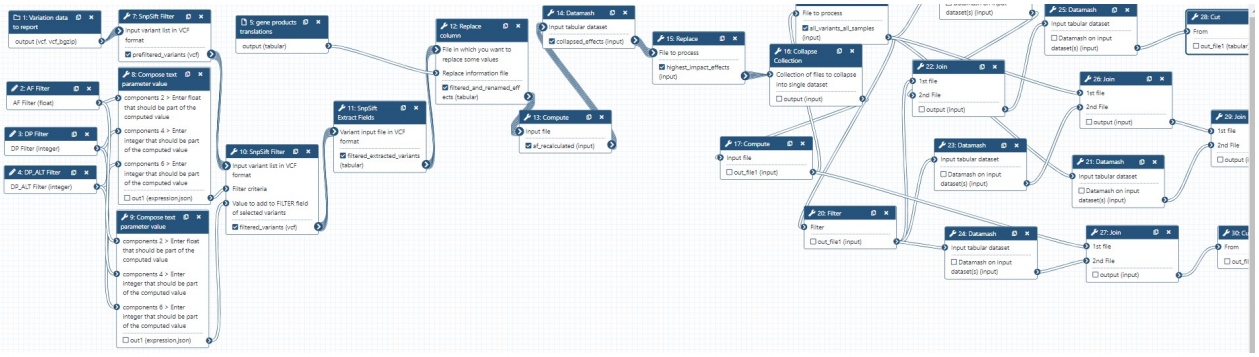

Figure 5.4: Segment of COVID-19: variation analysis reporting workflow (Adapted from [75])

The workflow has 6 inputs and 36 steps to generate the output. If scientists need to compose this workflow, it will be very complex and tedious for them. But if scientists can use a recommendation system to suggest the tool(s) for completing steps, they can avoid the complexities and save time. You may consider reusing workflows from various workflow repositories, but we checked that less than 50 % of workflows are reusable without any issues. A recommendation system for suggesting tool(s) can be a solution to these challenges.

5.5 Study Methodology

We first explore several SWfMSs, such as Taverna, Galaxy, CWL, Nextflow, KNIME, and Pegasus, and their available published workflows in different workflow repositories. By exploring and researching these SWfMSs, we discover that Galaxy is widely considered the best SWfMS due to its scalability and wide range of features. It allows users to easily manage complex workflows, create custom tasks, and monitor the progress of existing ones. Additionally, it has a user-friendly interface, making it easy for both novices and experienced users alike

to learn how to use the software quickly. Galaxy has a vibrant community to enable reproducible research in data-rich sciences and beyond. The community is located all around the world. They arrange community conferences and conduct mentor networks to build members in their community. Galaxy has interacted with many interesting communities like Conda and Bioconda, Open Life Science, Outreachy, Well-being and Mental Health Guideline, F1000Research, and so on [64]. So for conducting this study, we select Galaxy SWfMS-related workflows.

5.5.1 Data collection

For this study, we first explore all the available workflow repositories, and then we collect workflows from Galaxy, Galaxy Europe, and Galaxy Australia repositories. These repositories are publicly available. We also collect workflows from WorkflowHub, myExperiment, and GitHub. They contain workflows for various SWfMSs, but for conducting our study, we only select the Galaxy-related workflows.

Galaxy is an open-source platform that aims to FAIR data analysis and enables usage of tools from various domains, manages data by sharing and publishing, and enables reproducibility. Galaxy has 169 servers; among them, useGalaxy servers are publicly available. These are Galaxy Main, Galaxy Europe, and Galaxy Australia. We collect workflows and workflows histories from all of these repositories.

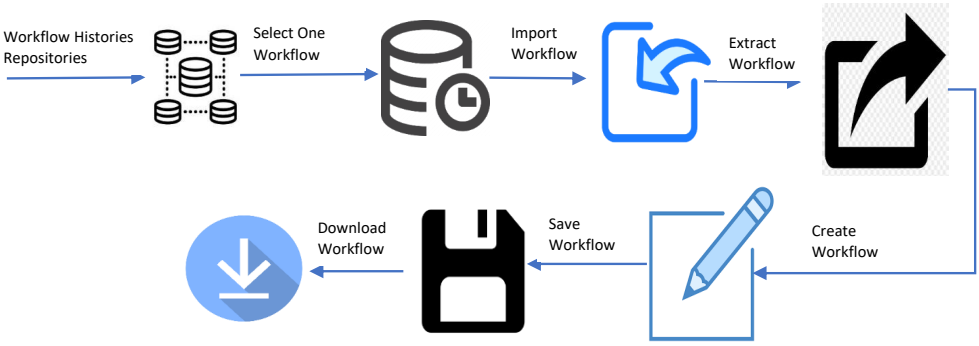

Figure 5.5: Workflow extraction process from workflow histories

Table 5.1 shows a distribution of workflows and workflow histories (December 15, 2022) for Galaxy SWfMS. We also collect Galaxy-related workflows from GitHub [105] and from Galaxy Training materials [73].

We can download the workflows from the workflow list of every repository. But for workflow histories, we need to import the workflows in our history (you need to be logged in to import the history); then, from the history, we need to extract the workflow, and then you need to click the Create button, and then save it. Now the workflow is ready for downloading. The whole process is depicted in Figure 5.5.

Table 5.1: Galaxy workflows and workflows histories distribution

Repository	No. of Workflows	No. of Workflow Histories
Galaxy Main	903	3236
Galaxy Europe	817	2700
Galaxy Australia	322	298
MyExperiment	79	0
Workflow Hub	129	0

5.5.2 Data preprocessing

First, we import all the Galaxy's workflows from the available sources. Later we check the workflow name and annotation for each workflow for Galaxy workflow repositories (we consider other sources, such as WorkflowHub, and Galaxy Training Workflows have proper names and annotations). We discard workflows where we do not get a proper name or annotation. We get nearly 7% such workflows in all our selected workflows. Therefore, we eliminate 7% of workflows in this way. Then, we find the workflows containing obsolete tools from the remaining workflows. We find 12.38% of workflows where one or more modules have outdated tools. As these tools are unavailable in Galaxy, we delete these modules from the workflows. We also discard incompatibly connected modules. Then we update/upgrade the versions of the tools wherever necessary. Initially, we had nearly nineteen thousand workflows. After eliminating error-containing, obsolete, and improper named workflows, we collected more than thirteen thousand workflows. These workflows are from different scientific analyses such as metagenomics, proteomics, RNA-seq, Hi-C, and assembly in the different Galaxy servers.

We collect the workflow's latest update time from the workflows sample, which tool is connected with another tool, and input and output tool versions. We discard the connection of the tool where the tools are obsolete. We also consider the tools' usage patterns. It is essential to analyze the usage pattern of tools because the recommended tools should have high relevance to the analysis. However, a high-usage tool can also be obsolete or may have been previously used a lot, but now its usage has become low. To check this issue, we also consider a cut-off date.

5.5.3 Data description

Galaxy supports a wide range of domains for performing various sorts of analysis like genomic file manipulation, bioinformatics, proteomics, genomic analysis, metagenomics, statistics and visualization, machine learning, and many other analytical works. Each domain has its own set of tools that can be used to achieve specific goals within the workflow. In addition to these, Galaxy also supports the integration of external data sources and databases, such as the UCSC Genome Browser and the Ensemble database, into workflows. Overall, Galaxy is a versatile SWFMS that can handle diverse datasets and support a wide range of scien-

tific research in the life sciences. We collect workflows from all the domains and obtain more than thirteen thousand workflows. On average, we found 18 tools for each workflow. Some workflows have only two tools, and a few have more than hundreds of tools.

5.6 Implementation of the Proposed Recommendation System

Typically, workflows can be presented as a directed acyclic graph where modules (tools) are connected sequentially. A workflow can have many tool sequences to analyze scientific data. A tool can consume a single dataset or a collection of datasets as input and produce one or multiple datasets as output. We collect the Galaxy SWfMS workflows from available workflow repositories to create the tool(s) recommendation system. We discard many workflows due to errors and other challenges. In Galaxy, a workflow may contain one or many tool sequences where tools are connected one after another sequentially.

For implementing the recommendation system, we constructed tool sequences from each workflow. A workflow is divided into smaller tool sequences, as shown in Figure 5.6. The last tool, shown as green, is the label of the subsequence, shown as blue. The label is the output, and it is learned and predicted by the recommendation system.

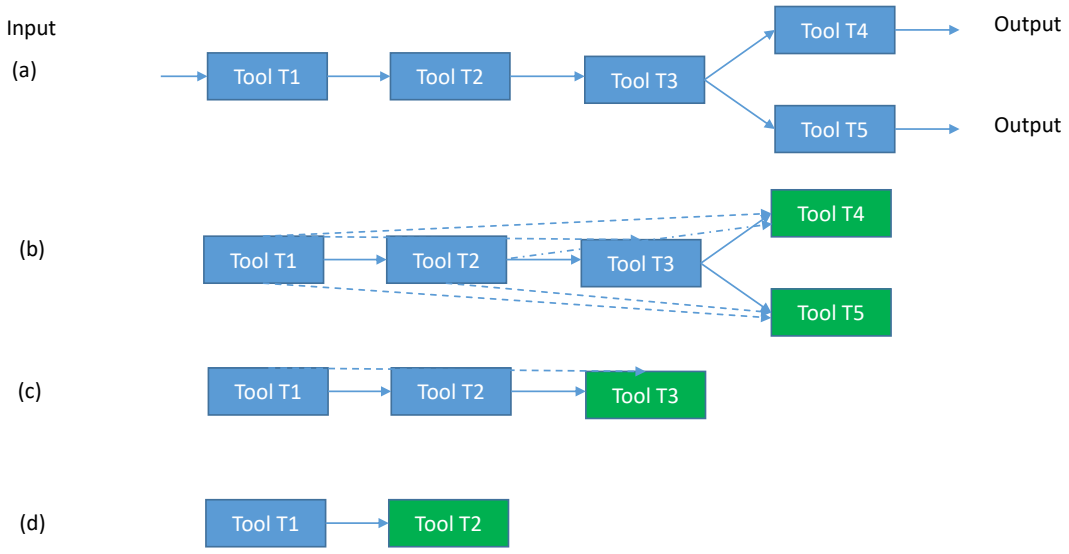

Figure 5.6: Tool sequence generation process

For Figure 5.6(b), Tool T4 and T5 are the labels of the subsequence Tool T1→Tool T2→Tool T3. A tool is not only dependent on its immediate predecessor but also on all prior tools in the tool sequence. For example, in Figure 5.6(c), tool T3 is dependent on tools T1 and T2. By analyzing multiple workflow fragments, the neural network learns that the label of a tool sequence Tool T1→Tool T2 is Tool T3.

The gated recurrent neural network is used in our implementation. We adapted the implementation approaches from Kumar et al. [106]. The key difference between our approach to their approach is they considered all of the workflows (error-containing, irrelevant, misleading...) while building the recommendation system. But in our approaches, we discard them so that our system can predict the most desirable tool(s).

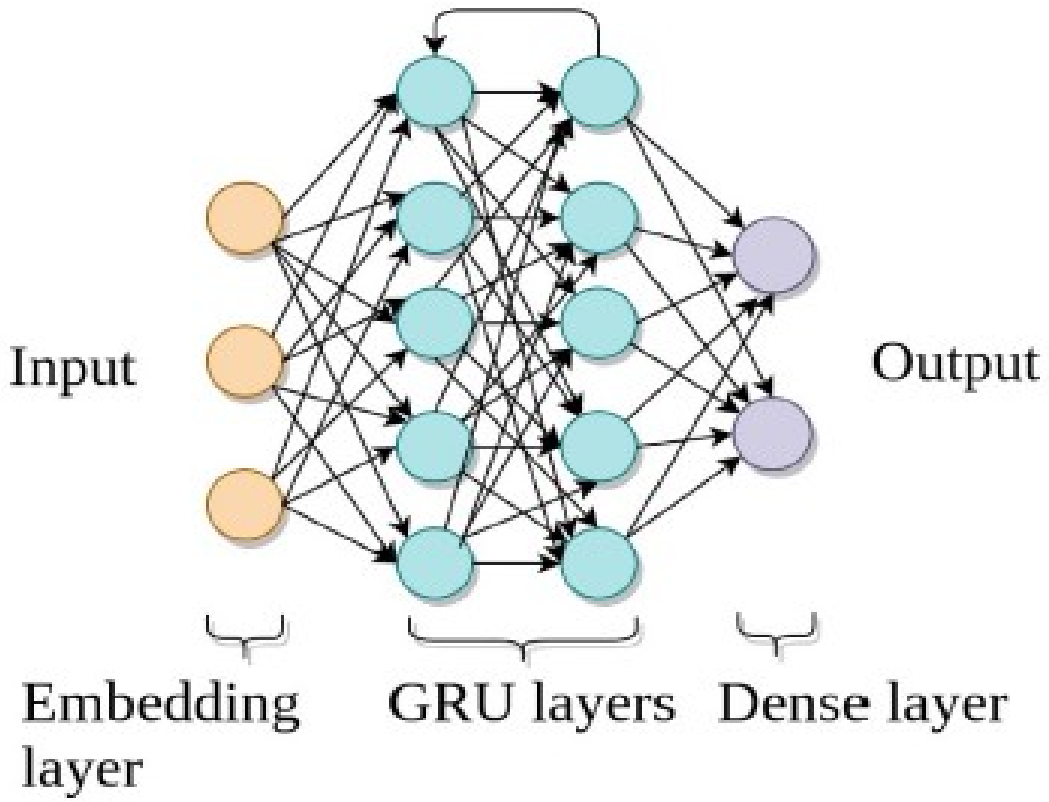

Figure 5.7: Architecture of GRU neural network (Adapted from [106])

Figure 5.7 shows the architecture of the GRU neural network, and we obtain it from Kumar et al. [106]. The neural network architecture has multiple components. The first component of the neural network is an input layer. The input layer learns embedding for each tool. The neural network uses a fixed-size vector as an internal representation of a tool. The tool sequences are transformed into input vectors as neural networks require input data as vectors and matrices.

We transform the tool sequence into a tool sequence vector, as shown in Figure 5.8(b), and a label vector, as shown in Figure 5.8(d). To form these vectors, we use a dictionary of tools (a dictionary is created using all the unique tools). Using the indices of tools, we create a tool sequence vector keeping the original order of tools. For example, if Tool T1 has an index of 17 in the dictionary, it is replaced by 17 in the vector. Our vector size is 25 (maximum tool sequence length 25). We padded the vector with trailing zeros to keep the length of the vector the same. The class labels are transformed into a bit vector using multi-hot encoding. The bit vector and sequence vector form a training sample for the neural network. A pair of vectors are

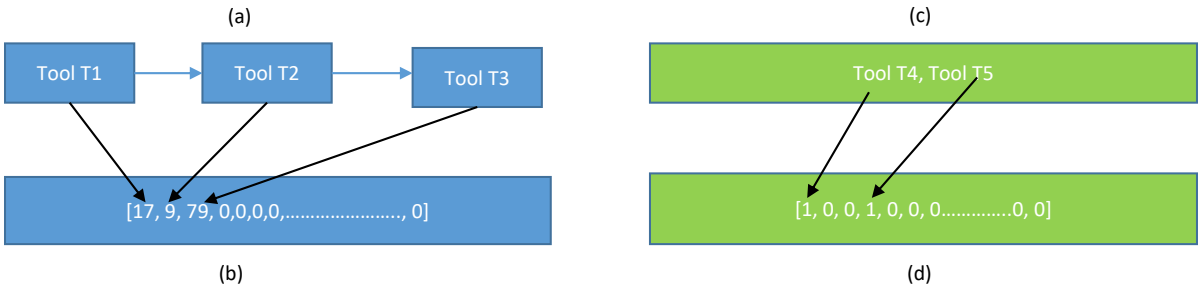

Figure 5.8: Tool sequence and its labels are transformed into vectors

created for each tool sequence. The matrices for tool sequences and respective labels form the input to the neural network.

The embedding is a fixed-size vector for each tool. This vector is used as an internal representation of a tool. The embedding vector replaces the indices of tools in each tool sequence. For example, the vector of a tool sequence [17, 9, 79, 0, 0, ..., 0] is transformed into [[0.5, 0.02, 0.007, ..., 0.27], [0.6, 0.2, 0.006, ..., 0.89], [...], 0, 0, ...,0] by the embedding layer. The same embedding vector represents a tool in all tool sequences in which the tool is present. The approach is similar to word embeddings in natural language processing [129].

The usage frequency of all tools to a certain cut-off date is collected and used in the training of the neural network as the weights of tools. If a tool has higher usage, it is assigned a higher weight. When tools are recommended, a score is assigned to each tool by the neural network. Highly used tools get higher weights. But it may happen that some tools that were used often in the past to create workflows are not used anymore. Therefore, assigning higher weights to these tools may not be a good indicator of their relevance. Using support vector regression, the trend of tool usage over time is predicted for the next month, and its logarithm is used as the weight for this tool in the neural network training. The logarithm of usage frequency is used because only a few tools have very high usage. Learning the trend for each tool involves 5-fold cross-validation and optimization of kernel and degree using grid search. We adopted the approaches from Kumar et al. [106].

Our next step is the GRU layers. GRU has several advantages that help it to learn sequential data. The traditional RNN suffers the problems of vanishing and exploding gradients, but GRU does not. On the other hand, GRU has fewer parameters than the long short-term memory network (LSTM), which makes using GRU simpler. The last component is the output layer. It predicts a score for each tool. The closer the predicted score is 1, the chances of being recommended are high. Overfitting occurs in a neural network when it performs well in training data, but performance on test data remains poor. We use dropout layers to minimize the effect of overfitting. Activation functions are used in a neural network to transform inputs to a layer into its outputs. We use two activations in our approach. One is an exponential linear unit (ELU), and another is a sigmoid function. A neural network learns patterns from the data by minimizing a loss function. For classification problems, cross-entropy is a popular choice for function; therefore, we use it in our approach. The hyper parameters in our system are optimized using Bayesian sequential model-based

optimization.

The training data contains 80% of all tool sequences, and test data contains the remaining 20%. The neural network learns patterns from training data in the tool sequences and generates a model. The training data are iterated over ten epochs of the neural network training. After learning from the training data, the generated model is used to predict tool(s). A probability score is assigned for each tool, and the recommended tools are sorted in descending order.

Figure 5.9 shows the user interface to see the recommended tool(s) in Galaxy. Users can predict tool(s) by using this system easily.

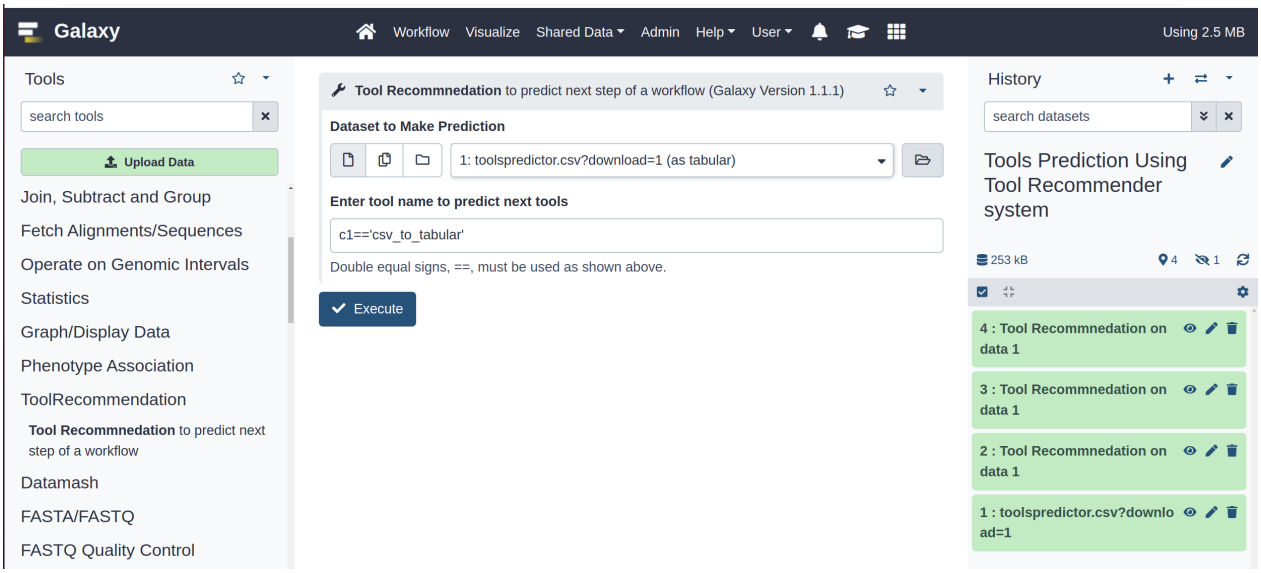

Figure 5.9: Integration of recommendation tool in Galaxy

5.7 Examples of Recommended Tools

To illustrate the real-time use of the recommendation system, we construct a workflow using the suggested tools for differentiating the different species of the Iris dataset and visualizing the Iris dataset features with two-dimensional scatterplots. For this, we need to create a history in Galaxy and name it as Exploring Iris dataset with statistics and scatterplots. We upload the iris.csv data file from Zenodo. We then need to convert the data into a tabular format for further processing. Then we use our recommendation system to suggest the next tools. We get *Remove beginning1, datamash_ops, tp_easyjoin_tool, collapse_dataset, cat1, Paste1, tp_awk_tool, datamash_transpose, addValue, tp_cat* as suggested tools. We select *Remove beginning1* and *datamash_ops* for our analysis. Then again, we search next compatibles tools and get *Grouping1, cat1, tp_cut_tool, comp1, ggplot2_point, Paste1, addName, Convert characters1, tp_awk_tool, cor2, mass_spectrometry_imaging_qc*. We choose the *tp_cut_tool, Grouping1 and, ggplot2_point* for our analysis. Following this way, we finish the workflow construction. Figure 5.10 shows the final workflow.

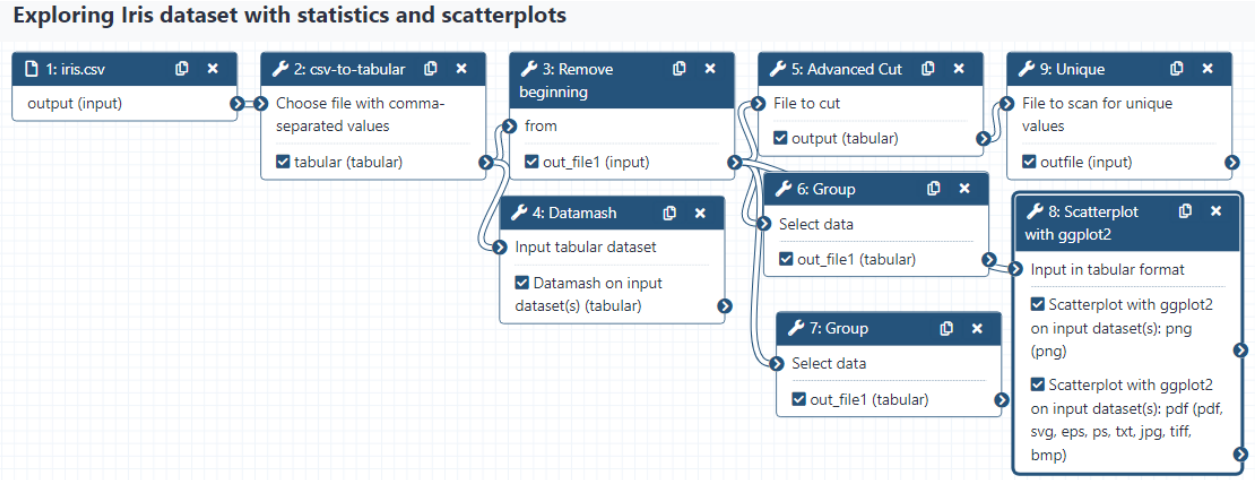

Figure 5.10: A workflow construction using suggested tools

Table 5.2 shows a list of recommended tools in different scientific analyses. Here we enlisted the recommended tools for freebayes, fastqc and Filter1.

Table 5.2: Recommended tools in scientific analyses

SL	Input Tool	Predicted Tools
1	freebayes	sickle, cat1, mothur_make_contigs, bwa_wrapper, fastq_to_fasta_python, spades, tringtie, rna_star,fastq_stats, fastqc, tophat2, hisat, hisat2, bwa, prinseq, snippy, multiqc, fastp, trimmer
2	fastqc	snpEff, fastq_to_fasta_python,stringtie, tophat2, Grep1, tp_cat, tp_head_tool, bwa, snpSift_annotate, multiqc, genomespace_exporter
3	Filter1	smooth_running_window, Convert characters1, random_lines1, tp_cut_tool, bedtools_mergebed, deg_annotate, collapse_dataset, wc_gnu, join_files_on_column_fuzzy

5.8 Evaluation

To evaluate a GRU-based recommendation system, we can use several metrics like *accuracy* (the proportion of items that the recommendation system correctly recommends), *precision* (the proportion of recommended items that are relevant to the user), *recall* (the proportion of relevant items that are recommended), *f1-score* (gives an overall score that represents the balance between precision and recall) and mean average precision (the average precision of the recommendation system for a set of users). *Precision@k* [21] is a widely used metric for evaluating recommendation systems because it provides information about the relevance of the top-k recommendations for a given user. If any recommendation model produces more data than people can process, then precision@k can be a perfect solution. It measures the proportion of relevant items in the top-k

recommendations out of all the items recommended to the user.

Precision@k is a metric used to evaluate the accuracy of sequential deep-learning models in predicting the correct labels for unseen data. It measures the fraction of true positives (TPs) among all returned results, with k representing the number of results returned by the model. For example, if a model returns three results and two out of those three are correct predictions, then precision@3 will be calculated as 2/3 or 0.67. We obtain the model to predict tool(s) using GRU neural network architecture. Our model recommends tool(s), and Top-k precision (precision@k) is one of the best metrics for evaluating a recommendation system [147, 100]. We use precision@1 and precision@2 metrics to evaluate the quality of the tool(s) recommendation system. The precision metrics are computed over ten training iterations for each experiment run, and we obtain 98% top-1 precision. We also obtain a good accuracy of 98% and a good recall value of 97% for our system.

In most real-world scenarios, the recommendation systems usually display the top-k recommendations to the users. Therefore, precision@k provides a realistic view as it focuses on the items that the users are most likely to interact with. In our system, we obtain 98% precision, which is promising.

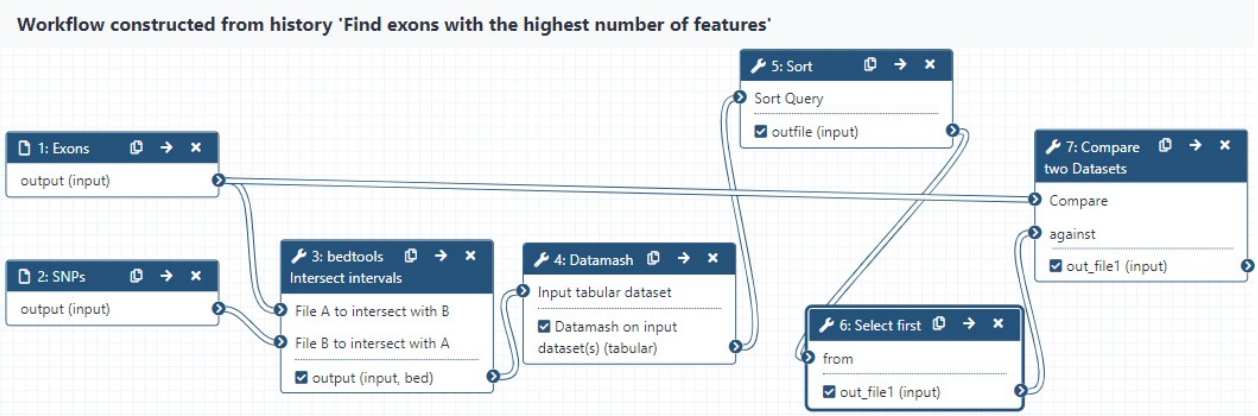

Figure 5.11: A workflow to find exon with the highest number of features

To evaluate the effectiveness of our tool(s) recommendation system, we construct a workflow to find exons with the highest number of features. We first fetch the data from Zenodo and rename the datasets to Exons and SNPs, respectively. Our target is to find which exon contains the most SNPs. So we need to intersect the file with exon locations with the file containing SNP locations. To find the intersection, we select *bedtools intersect intervals* from the tools section. Then our task is to count the number of SNPS per exon. So we use our system to predict the next compatible tools, and we get *Datamash, Filter, Cut, Join two datasets.....* Then our task is to sort the exons by SNPs count, and again, we use our system to predict the next tools, and we get *Sort, Select, Join...* as predicted tools. We use *Sort* for our operation. Following the similar way, we construct the workflow as shown in Figure 5.11.

We compare our workflow with the workflow provided by Galaxy trainer [159], and we notice that we generate the same workflow as the standard one. So we can say that our tool(s) recommendation system can

predict the appropriate tool(s) while constructing a workflow.

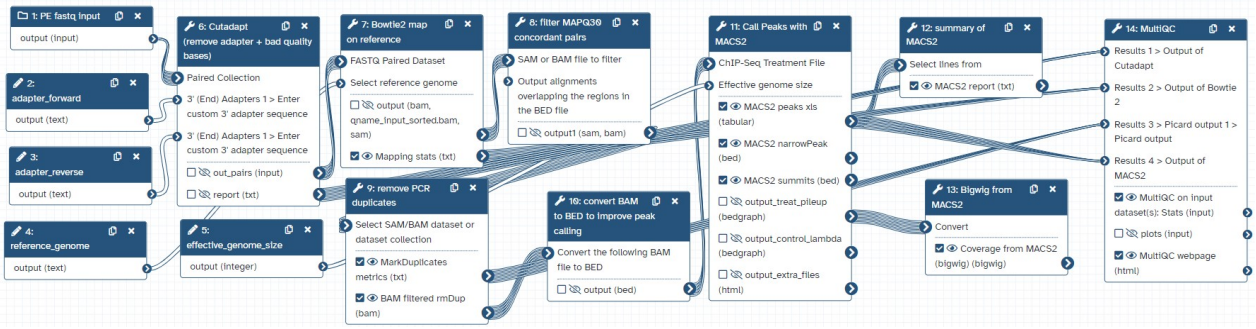

Figure 5.12: A workflow to have an overview of QC report of bioinformatics analyses

Let us consider another example of a workflow. The purposes of the workflow are to remove illumina adapters and low-quality bases and to have an overview of the QC using the MultiQC tool. The workflow has adapter sequences, reference_genome, and effective_genome_size as inputs and several tools like Cutadapt, Bowtie2, Filter SAM or BAM, etc., to produce the desired outcome. We construct the workflow by taking the help of our tool(s) recommendation system. The workflow is shown in Figure 5.12. We compare the workflow with the workflow provided by WorkflowHub [52], and we notice that our system can predict all the desired tool(s) to construct the workflow.

Using our tool(s) recommendation system, we also construct the workflow shown in 5.10. So we can tell that our system is able to recommend effective tool(s) to construct a workflow, and our system has a good precision value which is 98%.

5.9 Related Work

Reuse-oriented code recommendation systems support a large variety of common code reuse by generating the code using the knowledge gained by mining software source code repositories [97]. For large-scale software systems, refactoring operations are also supported by the recommendation system [11]. Repetitive software changes are now automatically possible by recommending program transformations [103]. Apart from these, recommendation systems are used in requirement discovery [89] and issue management [18] (changes, evolution, and bugs) in software development. Netflix [14], Amazon [94], YouTube [46], Alibaba [162], and most of the giant companies are using the benefits of recommendation systems effectively and efficiently. On the other hand, only a few recommendation systems are developed in the domain of scientific workflows. Researchers developed a few recommendation systems to simplify workflow construction and scientific analysis. Mass spectrometry-based proteomics [139] used EDAM and semantic annotations of tools to compose a workflow automatically. DiBernardo et al. [55] used data types to create a workflow automatically. Koop et al. [104] built *VisComplete*, a system to aid users in creating visualization pipelines based on VisTrails SWfMS using a database of previously created visualization pipelines but did not consider the correctness of the previous

pipelines. All these approaches apply to a limited set of bioinformatics analyses and are subject to error as the analysis tools become updated frequently. Kumar et al. [106] developed a model for suggesting tools while constructing a workflow using a deep learning approach by analyzing workflows available in the European Galaxy server. But they did not consider the type compatibility, tools annotations, errors, obsolete tools, and so on issues while suggesting the tools. To the best of our knowledge, our proposed approach is the first to consider previous workflow correctness to suggest tool(s) while constructing a new scientific workflow. Another important thing to mention here is that we collect workflows from multiple domains' scientific analyses to build the tool(s) recommendation system.

5.10 Limitations

In this study, we develop a tool(s) recommendation system using a gated recurrent neural network (GRU). While developing the system, we only consider workflows from Galaxy workflow repositories, so our system may not be compatible with other SWfMSs. Most of the workflows we used to construct the system are related to bioinformatics. Thus, our system may not be able to predict tool(s) effectively for other domains like earth science and chemistry. Another potential area for improvement with our implementation is the length of tool sequences. Our system can take a maximum of 25 tools in the input tool sequence. For evaluating the system, we construct a few workflows and compare them with the standard workflows sample and notice that our system can predict tool(s) successfully. But we do not conduct any user study to get feedback from them to use the system. In the future, we plan to make the system compatible with other SWfMSs and conduct a user study to check the system's effectiveness. We also have a plan to work with the length of tool sequences using transformers.

5.11 Conclusion and Future Work

For this study, we explore various workflow repositories, extract workflows, and identify several issues in the existing workflows. We also explore state-of-the-art tools and techniques for recommending workflows in scientific workflows and discover that none of them considered existing workflows' correctness in building their systems. So, we plan to work on it. As designing a scientific workflow is not an easy task but vital for many scientific domains, the tool(s) recommendation system should be helpful for scientists who are inexperienced in creating a workflow. The system only shows a few tools from a big collection of available tools in the Galaxy tool shed repository. Our future plan is to expand our tool(s) for other SWfMSs to suggest tool(s) while creating workflows based on error-free reusable workflows.

6 Conclusion and Future Work

This chapter concludes our thesis. First, we present a summary of the thesis in Section 6.1, and then Section 6.2 outlines future research directions from this thesis.

6.1 Summary

SWfMSs have emerged as valuable sources for large-scale scientific data analysis with numerous facilities for domain scientists with little or no programming knowledge. Among SWfMSs, Galaxy is one of the most useful platforms. It has several workflow repositories. The repositories accumulate thousands of scientific workflows from various domains ranging from bioinformatics to deep learning. The number of workflows shared at the repositories has steadily increased every month. The number of users in Galaxy and its community forum is also growing day by day. To assist the domain scientist and user community, we thus conducted three studies. In particular, we assess the reusability challenges of scientific workflows, find out the challenges of provenance management and also illustrate the way to construct the workflow smoothly and efficiently.

In our first study, we manually analyze 307 randomly selected scientific workflows from the Galaxy workflow repository and investigate their reusability challenges. We classify reusability status into two major categories, namely reusable and nonreusable. We can reuse about 42.35% of workflows without facing any obstacles. On the other hand, 30.94 % of workflows require minor or major modifications to make them reusable. We list the challenges of reusability. Then, we explain the ways to overcome the challenges with examples. Such challenges and our action list will help workflow composers create reusable workflows.

At different phases of scientific workflow life cycles, SWfMSs capture provenance information allowing result reproducibility, sharing, and knowledge reuse in the scientific community. Generally, this provenance information is many times larger than the workflow and its input dataset. Handling these exponentially increasing data volumes is an essential field that requires further investigation. Thus, in our second study, we discover the challenges of state-of-the-art provenance-capturing mechanisms. We also propose several approaches to overcome the challenges of provenance and efficient reuse of existing provenance data.

In our third study, we concentrate on making workflow construction simple for domain scientists to make their complex data-intensive analysis reliable, efficient, and easy-going. We collect Galaxy workflows from available workflow repositories, and using them; we construct a machine-learning model with the help of a gated recurrent neural network to predict compatible tool(s) for workflow construction. While building the model, we consider the workflow name, workflow correctness, usage frequencies, and annotation of the

workflow. Our model can predict the most expected tool(s) among the thousands of tools. Furthermore, our recommendation system can relieve scientists from memorizing and searching vast amounts of tools while constructing a workflow.

6.2 Future Work

In this thesis, for reusability challenges and tool(s) recommendation system, we mainly focus on Galaxy SWfMS. In the future, we have plans to extend these works with other popular SWfMSs. We discuss our future plans with the research work in the thesis below.

6.2.1 Exploring and mining of scientific workflows repositories

Open science has emerged as a framework for improving the quality of scientific analysis. Transparent, accessible knowledge-sharing, and collaborative networks are essential components of open science. Scientific workflows communities are also tending toward open science and, as a result, made many workflows and datasets available to the community in different repositories. The future of scientific advancements mostly depends on the ability of scientists to comprehend the vast amount of data currently being produced and acquired. But the repositories contain a significant number of nonreusable, duplicate workflows. Also, a lot of workflows are inefficient and ineffective. Finding out irrelevant, duplicate workflows and making the repositories intuitive to use can make the workflow developer's life more smooth. Thus, we have a plan to explore and mine scientific workflow repositories.

6.2.2 Benchmark dataset for reusable workflows

Reusability is one of the core aspects of any scientific data analysis. In our first study, we investigate the reusability challenges of scientific workflows and then offer an action list to overcome the difficulties for Galaxy SWfMS. In the future, we have plans to explore other available repositories and make the nonreusable workflows reusable by performing necessary actions. Then we will create a benchmark of the reusable workflow for several SWfMSs.

6.2.3 Enhancing provenance capabilities for SWfMSs

In our second study, we explore state-of-the-art provenance models and data-capturing mechanisms, discover their challenges, and propose ways to overcome them. In the future, we will build a provenance data storing system that will be compatible with any SWfMS and ensure optimized storing and reusing of provenance data along with reproducing results from the previous executions.

6.2.4 Suggesting analysis pipeline(s) for scientific datasets

Open science policy is becoming more popular among researchers, and scientists are now more open to publicly sharing their datasets and analysis pipelines so that others can benefit from using them. As the number of high-quality public datasets and analysis pipelines is increasing, we can use machine-learning approaches to predict analysis pipeline(s) based on datasets. Thus, we plan to build a pipeline(s) recommendation system based on the input datasets.

6.2.5 Challenges in SWfMSs development: A study of Stack Overflow posts and GitHub issues

SWfMSs are becoming increasingly popular due to their benefits in saving costs, time, and effort. SWfMSs development requires special expertise that differs from the traditional software systems. At the same time, the challenges SWfMS developers face remain mostly unknown since most studies focus on traditional software development. Therefore, we plan to examine the Q&A website, stack overflow, and open-source software repository, GitHub, to provide insights into the topics that SWfMS developers are interested in and the challenges they face.

References

[1] Alex Abramovici, William E Althouse, Ronald WP Drever, Yekta Gürsel, Seiji Kawamura, Frederick J Raab, David Shoemaker, Lisa Sievers, Robert E Spero, Kip S Thorne, et al. Ligo: The laser interferometer gravitational-wave observatory. *science*, 256(5055):325–333, 1992.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[3] Aymen Al-Saadi, Dong H Ahn, Yadu Babuji, Kyle Chard, James Corbett, Mihael Hategan, Stephen Herbein, Shantenu Jha, Daniel Laney, Andre Merzky, et al. Exaworks: Workflows for exascale. In *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pages 50–57. IEEE, 2021.

[4] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.*, pages 423–424. IEEE, IEEE, 2004.

[5] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In *International Provenance and Annotation Workshop*, pages 118–132. Springer, 2006.

[6] Manish Kumar Anand, Shawn Bowers, and Bertram Ludäscher. Techniques for efficiently querying scientific workflow provenance graphs. In *EDBT*, volume 10, pages 287–298, 2010.

[7] Annonymous. Reusability challenges of scientific workflows, 2023. URL `https://figshare.com/projects/ReusabilityChallengesofScientificWorkflows/157653`. Online; Last accessed January 2023.

[8] Ákos Balaskó. Workflow concept of ws-pgrade/guse. In *Science Gateways for Distributed Computing Infrastructures*, pages 33–50. Springer, 2014.

[9] Roger S Barga and Luciano A Digiampietri. Automatic capture and efficient storage of e-science experiment provenance. *Concurrency and Computation: Practice and Experience*, 20(5):419–429, 2008.

[10] Adam Barker and Jano van Hemert. Scientific workflow: a survey and research directions. In *International Conference on Parallel Processing and Applied Mathematics*, pages 746–753. Springer, 2007.

[11] Gabriele Bavota, Andrea De Lucia, Andrian Marcus, and Rocco Oliveto. Recommending refactoring operations in large software systems. In *Recommendation Systems in Software Engineering*, pages 387–419. Springer, 2014.

[12] Khalid Belhajjame, Katy Wolstencroft, Oscar Corcho, Tom Oinn, Franck Tanoh, Alan William, and Carole Goble. Metadata management in the taverna workflow system. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 651–656. IEEE, 2008.

[13] Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, et al. Prov-dm: The prov data model. *W3C Recommendation*, 14:15–16, 2013.

[14] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

[15] G Bruce Berriman, Ewa Deelman, John C Good, Joseph C Jacob, Daniel S Katz, Carl Kesselman, Anastasia C Laity, Thomas A Prince, Gurmeet Singh, and Mei-Hu Su. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *Optimizing scientific return for astronomy through information technologies*, volume 5493, pages 221–232. SPIE, 2004.

[16] Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Characterization of scientific workflows. In *2008 third Workshop on Workflows in Support of Large-scale Science*, pages 1–10. IEEE, 2008.

[17] Avijeet Biswal. Recurrent neural network(rnn) tutorial, 2022. URL `https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn`. Online; Last accessed November 2022.

[18] Markus Borg and Per Runeson. Changes, evolution, and bugs. In *Recommendation Systems in Software Engineering*, pages 477–509. Springer, 2014.

[19] Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys (CSUR)*, 37(1):1–28, 2005.

[20] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[21] Ryan Brideau. Precision@k: The overlooked metric for fraud and lead scoring models, 2022. URL `https://towardsdatascience.com/precision-k-the-overlooked-metric-for-fraud-and-lead-scoring-models-fabad2893c01`. Online; Last accessed November 2022.

[22] Peter Buneman and Wang-Chiew Tan. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1171–1173, 2007.

[23] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 539–550, 2006.

[24] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(Supplement 32):175–186, 2000.

[25] Anila Sahar Butt and Peter Fitch. Provone+: a provenance model for scientific workflows. In *International Conference on Web Information Systems Engineering*, pages 431–444. Springer, 2020.

[26] Anila Sahar Butt and Peter Fitch. A provenance model for control-flow driven scientific workflows. *Data & Knowledge Engineering*, 131:101877, 2021.

[27] Anila Sahar Butt, Nicholas J Car, and Peter Fitch. Towards ontology driven provenance in scientific workflow engine. In *MODELSWARD*, pages 105–115, 2020.

[28] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 745–747, 2006.

[29] Henri Casanova, Ewa Deelman, Sandra Gesing, Michael Hildreth, Stephen Hudson, William Koch, Jeffrey Larson, Mary Ann McDowell, Natalie Meyers, John-Luke Navarro, et al. Emerging frameworks for advancing scientific workflows research, development, and education. In *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pages 74–80. IEEE, 2021.

[30] Earthquake Center. Southern california earthquake center. *Caltech. Dataset*, 2013.

[31] Debasish Chakroborti, Manishankar Mondal, Banani Roy, Chanchal K Roy, and Kevin A Schneider. Optimized storing of workflow outputs through mining association rules. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 508–515. IEEE, 2018.

[32] Debasish Chakroborti, Banani Roy, and Sristy Sumana Nath. Designing for recommending intermediate states in a scientific workflow management system. *Proceedings of the ACM on Human-Computer Interaction*, 5(EICS):1–29, 2021.

[33] Artem Chebotko, Xubo Fei, Shiyong Lu, and Farshad Fotouhi. Scientific workflow provenance metadata management using an rdbms-based rdf store. *Wayne State University, Tech. Rep. TR-DB-092007-CFLF*, 2007.

[34] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[35] Edgar F Codd. Further normalization of the data base relational model. *Database Systems*, 6:33–64, 1972.

[36] Edgar F Codd. Recent investigations in relational data base systems. *Database Systems*, 1975.

[37] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[38] Sarah Cohen-Boulakia and Ulf Leser. Search, adapt, and reuse: the future of scientific workflows. *ACM SIGMOD Record*, 40(2):6–16, 2011.

[39] Flavio Costa, Vítor Silva, Daniel De Oliveira, Kary Ocaña, Eduardo Ogasawara, Jonas Dias, and Marta Mattoso. Capturing and querying workflow runtime provenance with prov: a practical approach. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 282–289, 2013.

[40] Daniela Soares Cruzes and Lotfi ben Othmane. Threats to validity in empirical software security research. *Empirical Research for Software Security: Foundations and Experience*, 2017.

[41] Vasa Curcin, Moustafa Ghanem, Patrick Wendel, and Yike Guo. Heterogeneous workflows in scientific workflow systems. In *International Conference on Computational Science*, pages 204–211. Springer, 2007.

[42] Sérgio Manuel Serra da Cruz, Patrícia M Barros, Paulo M Bisch, Maria Luiza Machado Campos, and Marta Mattoso. Provenance services for distributed workflows. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 526–533. IEEE, 2008.

[43] Sérgio Manuel Serra da Cruz, Maria Luiza M Campos, and Marta Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *2009 Congress on Services-I*, pages 259–266. IEEE, 2009.

[44] Rafael Ferreira da Silva, Rosa Filgueira, Ilia Pietri, Ming Jiang, Rizos Sakellariou, and Ewa Deelman. A characterization of workflow management systems for extreme-scale applications. *Future Generation Computer Systems*, 75:228–238, 2017.

[45] Rafael Ferreira da Silva, Loïc Pottier, Taina Coleman, Ewa Deelman, and Henri Casanova. Workflowhub: Community framework for enabling scientific workflow research and development. In *2020 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pages 49–56. IEEE, 2020.

[46] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296, 2010.

[47] Susan B Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1345–1350, 2008.

[48] Daniel De Oliveira, Eduardo Ogasawara, Fernanda Baião, and Marta Mattoso. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 378–385. IEEE, 2010.

[49] Daniel De Oliveira, Kary ACS Ocana, Eduardo Ogasawara, Jonas Dias, Joao Gonçalves, Fernanda Baiao, and Marta Mattoso. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. *Future Generation Computer Systems*, 29(7):1816–1825, 2013.

[50] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[51] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[52] Lucille Delisle. Workflowhub cutandrun, 2022. URL `https://workflowhub.eu/workflows/395?version=3`. Online; Last accessed November 2022.

[53] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Mining usage patterns from a repository of scientific workflows. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 152–157, 2012.

[54] Jonas Dias, Gabriel Guerra, Fernando Rochinha, Alvaro LGA Coutinho, Patrick Valduriez, and Marta Mattoso. Data-centric iteration in dynamic workflows. *Future Generation Computer Systems*, 46: 114–126, 2015.

[55] Michael DiBernardo, Rachel Pottinger, and Mark Wilkinson. Semi-automatic web service composition for the life sciences using the biomoby semantic web framework. *Journal of Biomedical Informatics*, 41 (5):837–847, 2008.

[56] Dockstore-Galaxy. Community workflow for galaxy, 2022. URL `https://dockstore.org/organizations`. Online; Last accessed August 2022.

[57] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[58] Thomas Fahringer, Radu Prodan, Rubing Duan, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, Hong-Linh Truong, Alex Villazon, and Marek Wieczorek. Askalon: A grid application development and computing environment. In *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.*, pages 10–pp. IEEE, 2005.

[59] Thomas Fahringer, Radu Prodan, Rubing Duan, Jüurgen Hofer, Farrukh Nadeem, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, Hong-Linh Truong, et al. Askalon: A development and grid computing environment for scientific workflows. In *Workflows for e-Science*, pages 450–471. Springer, 2007.

[60] Ian Foster, Jens Vockler, Michael Wilde, and Yong Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Proceedings 14th International Conference on Scientific and Statistical Database Management*, pages 37–46. IEEE, 2002.

[61] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[62] James Frew, Dominic Metzger, and Peter Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 20(5):485–496, 2008.

[63] Jeremy Frey, David De Roure, Kieron Taylor, Jonathan Essex, Hugo Mills, and Ed Zaluska. Combechem: a case study in provenance and annotation using the semantic web. In *International Provenance and Annotation Workshop*, pages 270–277. Springer, 2006.

[64] Galaxy. Interacting communities with galaxy, 2022. URL `https://galaxyproject.org/community/`. Online; Last accessed November 2022.

[65] Galaxy. Galaxy au shared histories., 2022. URL https://usegalaxy.org.au/histories/list_published. Online; Last accessed August 2022.

[66] Galaxy. Galaxy scientific workflow management system australia server published workflows., 2022. URL https://usegalaxy.org.au/workflows/list_published. Online; Last accessed August 2022.

[67] Galaxy. Galaxy community., 2022. URL https://galaxyproject.org/community/. Online; Last accessed August 2022.

[68] Galaxy. Galaxy community help forum., 2022. URL https://help.galaxyproject.org/. Online; Last accessed July 2022.

[69] Galaxy. Galaxy eu shared histories., 2022. URL https://usegalaxy.eu/histories/list_published. Online; Last accessed August 2022.

[70] Galaxy. Galaxy scientific workflow management system europe server published workflows., 2022. URL https://usegalaxy.eu/workflows/list_published. Online; Last accessed August 2022.

[71] Galaxy. Galaxy scientific workflow management system europe, 2022. URL https://usegalaxy.eu/. Online; Last accessed August 2022.

[72] Galaxy. Galaxy scientific workflow management system, 2022. URL https://usegalaxy.org/. Online; Last accessed April 2022.

[73] Galaxy. Galaxy training, 2022. URL https://training.galaxyproject.org/training-material/topics/introduction/. Online; Last accessed August 2022.

[74] Galaxy. Galaxy main shared histories., 2022. URL https://usegalaxy.org/histories/list_published. Online; Last accessed August 2022.

[75] Galaxy. Galaxy scientific workflow management system main server published workflows., 2022. URL https://usegalaxy.org/workflows/list_published. Online; Last accessed August 2022.

[76] Daniel Garijo, Pinar Alper, Khalid Belhajjame, Oscar Corcho, Yolanda Gil, and Carole Goble. Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems*, 36:338–351, 2014.

[77] Marko Gasparic and Andrea Janes. What recommendation systems for software engineering recommend: A systematic literature review. *Journal of Systems and Software*, 113:101–113, 2016.

[78] Belinda Giardine, Cathy Riemer, Ross C Hardison, Richard Burhans, Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert, James Taylor, et al. Galaxy: a platform for interactive large-scale genome analysis. *Genome Research*, 15(10):1451–1455, 2005.

[79] Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro Gonzalez-Calero, Paul Groth, Joshua Moody, and Ewa Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72, 2010.

[80] Carole A Goble, Jiten Bhagat, Sergejs Aleksejevs, Don Cruickshank, Danius Michaelides, David Newman, Mark Borkum, Sean Bechhofer, Marco Roos, Peter Li, et al. myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 38(suppl_2):W677–W682, 2010.

[81] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):1–13, 2010.

[82] Jennifer Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In *International Provenance and Annotation Workshop*, pages 101–108. Springer, 2006.

[83] Jennifer Golbeck and James Hendler. A semantic web approach to the provenance challenge. *Concurrency and Computation: Practice and Experience*, 20(5):431–439, 2008.

[84] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. Conventional workflow technology for scientific simulation. In *Guide to e-Science*, pages 323–352. Springer, 2011.

[85] Robert Graves, Thomas H Jordan, Scott Callaghan, Ewa Deelman, Edward Field, Gideon Juve, Carl Kesselman, Philip Maechling, Gaurang Mehta, Kevin Milner, et al. Cybershake: A physics-based seismic hazard model for southern california. *Pure and Applied Geophysics*, 168(3):367–381, 2011.

[86] Paul Groth, Simon Miles, Weijian Fang, Sylvia C Wong, K-P Zauner, and Luc Moreau. Recording and using provenance in a protein compressibility experiment. In *HPDC-14. Proceedings. 14th IEEE International Symposium on High Performance Distributed Computing, 2005.*, pages 201–208. IEEE, 2005.

[87] Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. An architecture for provenance systems. *"*, 2006.

[88] Arzu Tugce Guler, Cathelijn JF Waaijer, and Magnus Palmblad. Scientific workflows for bibliometrics. *Scientometrics*, 107(2):385–398, 2016.

[89] Negar Hariri, Carlos Castro-Herrera, Jane Cleland-Huang, and Bamshad Mobasher. Recommendation systems in requirements discovery. In *Recommendation Systems in Software Engineering*, pages 455–476. Springer, 2014.

[90] Sonja Holl, Olav Zimmermann, Magnus Palmblad, Yassene Mohammed, and Martin Hofmann-Apitius. A new optimization phase for scientific workflow management systems. *Future Generation Computer Systems*, 36:352–362, 2014.

[91] David A Holland, Uri Jacob Braun, Diana Maclean, Kiran-Kumar Muniswamy-Reddy, and Margo I Seltzer. Choosing a data model and query language for provenance. In *Proceedings of the 2nd International Provenance and Annotation Workshop (IPAW'08)*. Springer, 2008.

[92] David A Holland, Margo I Seltzer, Uri Braun, and Kiran-Kumar Muniswamy-Reddy. Passing the provenance challenge. *Concurrency and Computation: Practice and Experience*, 20(5):531–540, 2008.

[93] Muhammad M Hossain, Banani Roy, Chanchal K Roy, and Kevin A Schneider. Vizsciflow: A visually guided scripting framework for supporting complex scientific data analysis. *Proceedings of the ACM on Human-Computer Interaction*, 4(EICS):1–37, 2020.

[94] Folasade Olubusola Isinkaye, Yetunde O Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[95] iwc. Sars-cov-2-variant calling, 2022. URL `https://github.com/galaxyproject/iwc/tree/main/workflows/sars-cov-2-variant-calling`. Online; Last accessed November 2022.

[96] Vahid Jalili, Enis Afgan, Qiang Gu, Dave Clements, Daniel Blankenberg, Jeremy Goecks, James Taylor, and Anton Nekrutenko. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update. *Nucleic Acids Research*, 48(W1):W395–W402, 2020.

[97] Werner Janjic, Oliver Hummel, and Colin Atkinson. Reuse-oriented code recommendation systems. In *Recommendation Systems in Software Engineering*, pages 359–386. Springer, 2014.

[98] Malik Muhammad Junaid, Maximilian Berger, Tomas Vitvar, Kassian Plankensteiner, and Thomas Fahringer. Workflow composition through design suggestions using design-time provenance information. In *2009 5th IEEE International Conference on E-Science Workshops*, pages 110–117. IEEE, 2009.

[99] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, 29(3):682–692, 2013.

[100] Zhao Kang, Chong Peng, and Qiang Cheng. Top-n recommender system via matrix completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[101] Alexandros Kanterakis, Galateia Iatraki, Konstantina Pityanou, Lefteris Koumakis, Nikos Kanakaris, Nikos Karacapilidis, and George Potamias. Towards reproducible bioinformatics: the openbio-c scientific workflow environment. In *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 221–226. IEEE, 2019.

[102] Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 20(5):587–597, 2008.

[103] Miryung Kim and Na Meng. Recommending program transformations. In *Recommendation Systems in Software Engineering*, pages 421–453. Springer, 2014.

[104] David Koop, Carlos E Scheidegger, Steven P Callahan, Juliana Freire, and Cláudio T Silva. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.

[105] Anup Kumar. Galaxy eu workflows collection at github, 2022. URL `https://github.com/anuprulez/galaxy_tool_recommendation/tree/master/data`. Online; Last accessed August 2022.

[106] Anup Kumar, Helena Rasche, Björn Grüning, and Rolf Backofen. Tool recommender system in galaxy using deep learning. *GigaScience*, 10(1):giaa152, 2021.

[107] Chee Sun Liew, Malcolm P Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I Van Hemert. Scientific workflows: moving across paradigms. *ACM Computing Surveys (CSUR)*, 49(4): 1–39, 2016.

[108] Chunhyeok Lim, Shiyong Lu, Artem Chebotko, and Farshad Fotouhi. Prospective and retrospective provenance collection in scientific workflow environments. In *2010 IEEE International Conference on Services Computing*, pages 449–456. IEEE, 2010.

[109] Cui Lin, Shiyong Lu, Zhaoqiang Lai, Artem Chebotko, Xubo Fei, Jing Hua, and Farshad Fotouhi. Service-oriented architecture for view: a visual scientific workflow management system. In *2008 IEEE International Conference on Services Computing*, volume 1, pages 335–342. IEEE, 2008.

[110] Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai, Zhaoqiang Lai, Farshad Fotouhi, and Jing Hua. A reference architecture for scientific workflow management systems and the view soa solution. *IEEE Transactions on Services Computing*, 2(1):79–92, 2009.

[111] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[112] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. *Parallelization of scientific workflows in the cloud*. PhD thesis, INRIA, 2014.

[113] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13(4):457–493, 2015.

[114] Jonathan Livny, Hidayat Teonadi, Miron Livny, and Matthew K Waldor. High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas. *PloS one*, 3(9):e3197, 2008.

[115] R Logesh, V Subramaniyaswamy, D Malathi, N Sivaramakrishnan, and Varadarajan Vijayakumar. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. *Neural Computing and Applications*, 32(7):2141–2164, 2020.

[116] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender Systems Handbook*, pages 73–105, 2011.

[117] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[118] Eric Lyons and Michael Freeling. How to usefully compare homologous plant genes and chromosomes as dna sequences. *The Plant Journal*, 53(4):661–673, 2008.

[119] Philip Maechling, Ewa Deelman, Li Zhao, Robert Graves, Gaurang Mehta, Nitin Gupta, John Mehringer, Carl Kesselman, Scott Callaghan, David Okaya, et al. Scec cybershake workflows—automating probabilistic seismic hazard analysis calculations. In *Workflows for e-Science*, pages 143–163. Springer, 2007.

[120] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[121] Maher Malaeb. Recommendation system worklow, 2022. URL `https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54`. Online; Last accessed December 2022.

[122] Anderson Marinho, Leonardo Murta, Cláudia Werner, Vanessa Braganholo, Sérgio Manuel Serra da Cruz, Eduardo Ogasawara, and Marta Mattoso. Provmanager: a provenance management system for scientific workflows. *Concurrency and Computation: Practice and Experience*, 24(13):1513–1530, 2012.

[123] Phillip Mates, Emanuele Santos, Juliana Freire, and Cláudio T Silva. Crowdlabs: Social analysis and visualization for the sciences. In *International Conference on Scientific and Statistical Database Management*, pages 555–564. Springer, 2011.

[124] Marta Mattoso, Claudia Werner, Guilherme Horta Travassos, Vanessa Braganholo, Eduardo Ogasawara, Daniel Oliveira, Sergio Cruz, Wallace Martinho, and Leonardo Murta. Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management*, 5(1):79–92, 2010.

[125] Joseph A Maxwell. *Qualitative research design: An interactive approach*. Sage Publications, 2012.

[126] Mandana Mazaheri, Gregory Kiar, and Tristan Glatard. A recommender system for scientific datasets and analysis pipelines. In *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pages 1–8. IEEE, 2021.

[127] Prem Melville and Vikas Sindhwani. Recommender systems. *Encyclopedia of Machine Learning*, 1: 829–838, 2010.

[128] Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling sequential data using higher-order relational features and predictive training. *arXiv preprint arXiv:1402.2333*, 2014.

[129] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[130] Golam Mostaeen. *Towards Collaborative Scientific Workflow Management System*. PhD thesis, University of Saskatchewan, 2019.

[131] Golam Mostaeen, Banani Roy, Chanchal Roy, and Kevin Schneider. Designing for real-time groupware systems to support complex scientific data analysis. *Proceedings of the ACM on Human-Computer Interaction*, 3(EICS):1–28, 2019.

[132] myExperiment. Workflows in myexperiment., 2022. URL `https://www.myexperiment.org/workflows`. Online; Last accessed June 2022.

[133] Kary ACS Ocaña, Daniel de Oliveira, Eduardo Ogasawara, Alberto MR Dávila, Alexandre AB Lima, and Marta Mattoso. Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *Brazilian Symposium on Bioinformatics*, pages 66–70. Springer, 2011.

[134] Kary ACS Ocaña, Daniel de Oliveira, Jonas Dias, Eduardo Ogasawara, and Marta Mattoso. Discovering drug targets for neglected diseases using a pharmacophylogenomic cloud workflow. In *2012 IEEE 8th International Conference on E-Science*, pages 1–8. IEEE, 2012.

[135] Kary ACS Ocaña, Daniel de Oliveira, Felipe Horta, Jonas Dias, Eduardo Ogasawara, and Marta Mattoso. Exploring molecular evolution reconstruction using a parallel cloud based scientific workflow. In *Brazilian Symposium on Bioinformatics*, pages 179–191. Springer, 2012.

[136] Brian D O'Connor, Denis Yuen, Vincent Chung, Andrew G Duncan, Xiang Kun Liu, Janice Patricia, Benedict Paten, Lincoln Stein, and Vincent Ferretti. The dockstore: enabling modular, community-focused sharing of docker-based genomics tools and workflows. *F1000Research*, 6, 2017.

[137] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[138] Wellington Oliveira, Daniel De Oliveira, and Vanessa Braganholo. Provenance analytics for workflow-based computational experiments: A survey. *ACM Computing Surveys (CSUR)*, 51(3):1–25, 2018.

[139] Magnus Palmblad, Anna-Lena Lamprecht, Jon Ison, and Veit Schwämmle. Automated workflow composition in mass spectrometry-based proteomics. *Bioinformatics*, 35(4):656–664, 2019.

[140] Sung Yong Park, Gina Faraci, Pamela M Ward, Jane F Emerson, and Ha Youn Lee. High-precision and cost-efficient sequencing for real-time covid-19 surveillance. *Scientific reports*, 11(1):1–10, 2021.

[141] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.

[142] Ajinkya Prabhune, Aaron Zweig, Rainer Stotzka, Jürgen Hesser, and Michael Gertz. P-pif: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *Distributed and Parallel Databases*, 36(1):219–264, 2018.

[143] Galaxy Project. Genomespace importer/exporter broken on galaxy., 2022. URL `https://github.com/galaxyproject/galaxy/issues/5527`. Online; Last accessed August 2022.

[144] Galaxy Project. Galaxy-project tools statistics., 2022. URL `https://galaxyproject.org/galaxy-project/statistics/#tools`. Online; Last accessed June 2022.

[145] Martin Robillard, Robert Walker, and Thomas Zimmermann. Recommendation systems for software engineering. *IEEE Software*, 27(4):80–86, 2009.

[146] Satya S Sahoo, Amit Sheth, and Cory Henson. Semantic provenance for escience: Managing the deluge of scientific data. *IEEE Internet Computing*, 12(4):46–54, 2008.

[147] Alan Said, Alejandro Bellogín, and Arjen De Vries. A top-n recommender system evaluation protocol inspired by deployed systems. In *LSRS Workshop at ACM RecSys*. Citeseer, 2013.

[148] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10):e1003285, 2013.

[149] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36, 2005.

[150] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A framework for collecting provenance in data-centric scientific workflows. In *2006 IEEE International Conference on Web Services (ICWS'06)*, pages 427–436. IEEE, 2006.

[151] Ganesh Sivaraman, Nicholas E Jackson, Benjamin Sanchez-Lengeling, Álvaro Vázquez-Mayagoitia, Alán Aspuru-Guzik, Venkatram Vishwanath, and Juan J De Pablo. A machine learning workflow for molecular analysis: application to melting points. *Machine Learning: Science and Technology*, 1(2): 025015, 2020.

[152] Kamran Soomro, Kamran Munir, and Richard McClatchey. Incorporating semantics in pattern-based scientific workflow recommender systems: Improving the accuracy of recommendations. In *2015 Science and Information Conference (SAI)*, pages 565–571. IEEE, 2015.

[153] Divesh Srivastava and Yannis Velegrakis. Intensional associations between data and metadata. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 401–412, 2007.

[154] Johannes Starlinger, Bryan Brancotte, Sarah Cohen-Boulakia, and Ulf Leser. Similarity search for scientific workflows. *Proceedings of the VLDB Endowment (PVLDB)*, 7(12):1143–1154, 2014.

[155] Johannes Starlinger, Sarah Cohen-Boulakia, Sanjeev Khanna, Susan B Davidson, and Ulf Leser. Effective and efficient similarity search in scientific workflow repositories. *Future Generation Computer Systems*, 56:584–594, 2016.

[156] Julia Stoyanovich, Ben Taskar, and Susan Davidson. Exploring repositories of scientific workflows. In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science*, pages 1–10, 2010.

[157] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. The triana workflow environment: Architecture and applications. In *Workflows for e-Science*, pages 320–339. Springer, 2007.

[158] Kieron R Taylor, Robert J Gledhill, Jonathan W Essex, Jeremy G Frey, Stephen W Harris, and Dave C De Roure. Bringing chemical data onto the semantic web. *Journal of Chemical Information and Modeling*, 46(3):939–952, 2006.

[159] Galaxy Training. Introduction to galaxy analysis., 2022. URL `https://training.galaxyproject.org/training-material/topics/introduction/tutorials/galaxy-intro-101/tutorial.html`. Online; Last accessed December 2022.

[160] UCSC. Ucsc data file format of genomes., 2022. URL `https://genome.ucsc.edu/FAQ/FAQformat.html`. Online; Last accessed August 2022.

[161] W3C. The prov data model, 2022. URL `https://www.w3.org/TR/prov-dm/`. Online; Last accessed August 2022.

[162] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 839–848, 2018.

[163] Wendy A Warr. Scientific workflow systems: Pipeline pilot and knime. *Journal of Computer-aided Molecular Design*, 26(7):801–804, 2012.

[164] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):1–9, 2016.

[165] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[166] WorkflowHub. Covid-19: Variation analysis reporting, 2022. URL `https://workflowhub.eu/workflows/109?version=6`. Online; Last accessed November 2022.

[167] WorkflowHub. Workflows in workflowhub., 2022. URL `https://workflowhub.eu/workflows`. Online; Last accessed August 2022.

[168] Justin M Wozniak, Rajeev Jain, Prasanna Balaprakash, Jonathan Ozik, Nicholson T Collier, John Bauer, Fangfang Xia, Thomas Brettin, Rick Stevens, Jamaludin Mohd-Yusof, et al. Candle/supervisor: A workflow framework for machine learning applied to cancer research. *BMC bioinformatics*, 19(18): 59–69, 2018.

[169] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[170] Reng Zeng, Xudong He, and Wil MP van der Aalst. A method to mine workflows from provenance for assisting scientific workflow composition. In *2011 IEEE World Congress on Services*, pages 169–175. IEEE, 2011.

[171] Jia Zhang, Wei Tan, John Alexander, Ian Foster, and Ravi Madduri. Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse. In *2011 IEEE International Conference on Services Computing*, pages 48–55. IEEE, 2011.

[172] Jun Zhao, Chris Wroe, Carole Goble, Robert Stevens, Dennis Quan, and Mark Greenwood. Using semantic web technologies for representing e-science provenance. In *International Semantic Web Conference*, pages 92–106. Springer, 2004.

[173] Jun Zhao, Carole Goble, Robert Stevens, and Daniele Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 20(5):463–472, 2008.

[174] Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor Von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, and Michael Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In *2007 IEEE Congress on Services (Services 2007)*, pages 199–206. IEEE, 2007.