Capturing and semantically describing provenance to tell the story of R scripts

Maria Luiza Mondelli*, Sheeba Samuel†, Birgitta König-Ries†, Luiz M. R. Gadelha Jr.*†
*National Laboratory for Scientific Computing, Petrópolis - RJ, Brazil
{mluiza,lgadelha}@lncc.br
†Friedrich-Schiller University, Jena, Germany
{sheeba.samuel,birgitta.koenig-ries}@uni-jena.de

***Abstract*—Reproducibility is a topic that has received significant attention in recent years. Despite being considered a fundamental factor in the scientific process, recent surveys have shown the difficulty of reproducing already published works, which impacts scientists' ability to verify, validate, and reuse research findings. Recording provenance data is one of the approaches that can help to mitigate the challenges involved in the reproducibility process. When semantically well defined, provenance can describe the entire process involved in producing a given result. Additionally, the use of semantic web technologies can allow for the provenance data to be *machine-actionable*. With a focus on computational experiments, this work presents a package for collecting and describing provenance data from R scripts using the REPRODUCE-ME ontology to describe the path taken to produce results. We describe the package implementation process and demonstrate how it can help describe the story of experiments defined as R scripts to support reproducibility.**

***Index Terms*—Reproducibility, Provenance, Ontology, Computational experiments, R scripts**

I. Introduction

Often referred to as one of the most important pillars of scientific research, reproducibility concerns the ability to perform a method more than once and arrive at results and conclusions consistent with those originally obtained. The practice of reproducible research allows results to be validated, verified, and reused by third parties. This pillar alone does not determine the production of new scientific findings. It must be accompanied by scientific rigor and transparency. However, reproducibility supports the development of future works based on more solid foundations [1].

In recent years, reproducibility in scientific research has been a discussed topic in many different domains. Studies and surveys have pointed to the poor reproducibility of published works, leading to a reproducibility crisis in research [2], [3]. The reasons for this may vary depending on the domain, but part of them focuses on the lack of adoption of good practices, such as sharing the research objects involved. A Nature survey points out that other reasons include lack of incentives, the pressure to publish, and selective reporting [2].

With the increasing advancement and democratization of the computational infrastructure, such as the availability of cloud services, research groups have benefited from computational approaches to carry out their experiments and analyses. This encourages the development of new tools and methods as well as the production and availability of data to be processed. In this context, an experiment that uses computing most of the time comprises a set of steps performed by a range of applications (software) and involves consuming and producing data. Experiments in this format can execute a flow of steps, where the data generated by one step is consumed by the subsequent step.

One can then think that computational experiments are, in essence, easy to be reproduced: the execution of an application, or a flow of applications, with the same input data should generate the same results. However, this is not always the case, as not all steps are necessarily deterministic. Besides, minor variations in the various layers involved in executing these experiments, such as hardware, software, data, and code, can significantly impact the results [4]. This can be exacerbated in cases where experiments are data and compute-intensive.

Some recent approaches have been proposed and implemented to support, allow and encourage the practice of reproducibility in scientific research. Some examples include tools to facilitate the work of making the experiment reproducible by scientists, adjustments to publication policy in journals, checklists on relevant information to be reported in papers, and conference tracks that focus on reproducible works. In this work, we explore the use of an ontology that can support the process of making an experiment reproducible by allowing a better understanding of the processes involved in its execution. For this, we have developed a mechanism for collecting provenance data from experiments that use the R scripting language according to the REPRODUCE-ME data model [5]. We demonstrate and evaluate this approach through a set of competency questions and discuss how the ontology can support reproducibility. We also discuss how this approach can be integrated with a computational reproducibility framework we proposed previously [6].

Therefore, in Section II, we present a review of the literature on approaches to allow the reproducibility of computational experiments. In Section III, we describe the approach for semantic representation of the provenance of R scripts, and also present and evaluate a package developed for this purpose. Finally, in Section IV we present some concluding remarks.

II. Related Work

The challenge of supporting reproducibility is closely linked to the number of factors involved in the life cycle of computational experiments. Ivie et al. [4] present a survey of these factors, separating them into scopes, or layers, which include: the command needed to execute a script, data, software, operating system, kernel, and hardware. In this sense, some studies have pointed out good practices for the development of reproducible research [7], [8]. In general, they recommend: (i) the use of automation to execute the experiments, avoiding, for example, manual steps for data manipulation; (ii) keeping a record of all executed steps; (iii) code and data versioning; (iv) availability of code and data in public repositories; (v) preserving experiments in containers or virtual machines. It is increasingly common to see conferences and journals that recommend some of these good practices. Also, research domains have proposed checklists and guidelines to include some specifics according to the perceived needs [9]–[11].

Computational experiments usually involve the composition of tasks connected through data consumption and production. Workflow management systems (WfMS) can be considered a type of approach to automate these experiments [12]. Systems like this provide features for scalable execution, management, and tracking of data, applications and their dependencies, and fault tolerance techniques. Scripting languages are also another possibility for automation. Its use, especially in studies involving data science techniques, has been widespread. Languages like R and Python are well-established, general-purpose languages with a range of packages available, providing flexibility in defining workflow steps.

Being able to record what was done in the experiment is linked to the concept of provenance. From a computational point of view, provenance can be classified in two forms [13]: (i) prospective, which concerns the experiment format, for instance, through a script or workflow definition containing the steps that compose it; and (ii) retrospective, which includes details on the execution of these steps including, for example, what input data, parameters, and computational environment were used. In this sense, provenance can be considered an important aspect to support reproducibility, as it provides the means to describe the entire process that led to the generation of a result.

Approaches that use provenance to support reproducibility have been developed in recent years. *ReproZip* [14] is an example of a packaging tool that uses provenance to identify dependencies required to run an experiment, and it builds a container in which the user can re-run the experiment. Similarly, *encapsulator* [15], is a toolbox that relies on provenance data to produce an environment in which computational experiments can be reproduced. WfMS usually provides functionality for capturing and analyzing provenance data, and for scripting languages, we have noWorkflow [16] for collecting provenance of Python scripts, the drake [17] and RDataTracker [18] packages for R, and ProvBook [19] for the provenance of Jupyter notebooks.

The basis for a mechanism aimed at collecting and recording provenance data is the definition of how this data should be represented. In addition, the granularity level at which this should happen must also be taken into account: a higher level can consider workflow activities, their dependencies, and parameter settings while a lower level considers aspects of the computing environment such as memory consumption, processing time, and implicit/hidden library dependencies. One approach to the representation of the provenance model comprises the use of ontologies [20]. A first example is the OPM model [21], which models provenance as a graph where nodes can be artifacts, processes, or agents. Its creation was followed by the definition of the PROV model, which is a well-established model and used as a reference by other models, such as the P-Plan [22], D-PROV [23], ProvONE [24], and PROV-Wf [25]. These models introduced more specific aspects about workflows, workflow technologies, and the possibility of representing domain data. Including both computational and non-computational steps, the REPRODUCE-ME ontology extends PROV and P-Plan to describe a complete path of a scientific experiment. Its development was initially based on life science experiments, but it has been adjusted to cover aspects regardless of the domain.

Because REPRODUCE-ME allows for greater detail of aspects involved in experiments defined as workflows than, for example, the PROV model alone as it is the case of RDataTracker, we understand that it fits our goal of representing the provenance of computational experiments defined with R scripts. Furthermore, as we aim to support reproducibility, we need to have access to information that provides greater understandability and interoperability, and with that, the need for a vocabulary capable of detailing the research objects involved in computational experiments. To our knowledge, there is no tool that semantically describes the provenance of R scripts at this level of detail. Furthermore, we understand that because it is an ontology based on well-established models and can be extended and modularized, it allows us to use it as the foundation of a bigger framework to support reproducibility. Therefore, in this work, we evaluate the applicability of the ontology to represent the provenance of R scripts.

III. Towards provenance of R scripts

In previous work, we proposed a framework to enable the reproducibility of computational experiments, especially those that use scripting languages to define their workflows [6]. We suggested that this framework could follow, even minimally, the FAIR principles [26] so that the research artifacts involved were available and adequately identified. From this work, we identified the need to have a vocabulary capable of representing the details of the artifacts and layers involved in computational experiments to better adapt to the FAIR principles and provide broader support for reproducibility. In this work, we take a step in this direction, considering computational experiments that are defined by scripting languages, more specifically R, and applying the REPRODUCE-ME ontology to represent the provenance in greater detail semantically.

R is an open-source scripting language and environment for statistical and graphical methods, supported by the R Foundation for Statistical Computing and a large community of open-source developers and contributors. R can be considered a flexible language and environment since it can be extended with other functions beyond the functions available by default. These functions are distributed through external packages that can be installed and coupled to the environment. R users are encouraged to create their own packages and submit them to the Comprehensive R Archive Network (CRAN), the official R package repository. Currently, CRAN contains approximately 17600 published packages. Other packages can also be made available on platforms such as GitHub.

Because of its flexibility, R has been extensively used to construct and execute data science workflows and as a means of implementing routines for automating tasks that need to be executed frequently, for example. In addition, the use of scripting languages such as R has been encouraged by universities and research institutes, with the inclusion of classes and extension courses in their programs, in areas that are not necessarily those of technology or computational sciences. This is because different domains have seen the opportunity to add value to their work with the development of scripts, either to make them available or to improve their own productivity.

When implementing workflows in R, small tasks or activities comprise functions. Usually, functions receive input data, such as files or another type of object, and generate output data consumed by the following function. R functions are stored in packages, and the most commonly used R packages include features for manipulating, cleaning, and visualizing data. Functions encapsulate a set of smaller routines and are often defined to receive arguments for their parameters. These parameters may or may not have arguments with default values, and the user can perform the same function informing different arguments for the parameters in different moments. This is, in fact, a very common behavior, executing functions, or even entire scripts, with variations in their parameters. In the context of data science workflows, for example, these executions, or trials, can be made to identify the best configuration for adjusting a model considering a given input data.

The R environment does not automatically record provenance data from different executions of the same function or R script. This is one factor differentiating R and other scripting languages from some scientific workflow management systems that provide provenance management by default. However, because of the flexibility mentioned earlier, this functionality can be coupled to the environment by loading packages specifically designed for this purpose.

A. Semantic representation of R scripts with REPRODUCE-ME

Based on the overview of the tools available for collecting provenance in R and the most common ontologies used for this purpose, presented in Section 2, we opted for using the REPRODUCE-ME ontology to represent the provenance of R scripts. Therefore, we describe in more detail how some of the REPRODUCE-ME classes can be mapped to each of the components of an R script that we believe are the most relevant to be reported in the provenance data:

- Script: comprises the file containing the code that defines a particular workflow or experiment and is then part of the prospective provenance.
- Function: a set of encapsulated instructions aimed at executing a specific routine within a script. It may or may not consume and generate data, depending on how it was defined.
- Package: the component responsible for grouping various functionalities, such as functions. Users commonly use functions that have already been implemented and are made available through packages.
- Version (package): it concerns the version of the package. Implementations of functions within packages can be modified, so it is necessary to identify the version of the package loaded in a script. Small changes can impact results obtained between package versions.
- Programming language: it concerns the programming language in which the script is defined. In this work, we focus on R scripts, but as the ontology itself suggests, it can be applied to other languages.
- Version (programming language): it specifies the programming language version used to define and execute the R script. As with packages, variations between programming language versions can lead to variations in the final results.
- Operating system (OS): it describes the operating system used to run the R script.
- Version (operating system): it specifies the operating system version used to define and execute the R script.
- Operation: comprises R commands that perform arithmetic or assignment operations, which do not involve the execution of functions.
- Function activation: it concerns the execution of a function and is part of the prospective provenance. This is because the same function can be executed more than once and with different parameter settings. Also, as functions are executed, the order in which this happens or the chain between functions' inputs and outputs allows us to retrieve the script execution flow.
- Parameter: It concerns the input expected by the function for its execution/activation.
- Argument: it specifies the input value informed for a given parameter. Functions can have argument values set by default, but the user can modify them as needed. As changes in the input values of arguments can impact the results obtained, this is an essential provenance data to be recorded.
- Output: it concerns the result generated by the execution/activation of a function.

A diagram indicating how these classes connect to each other is presented in the Fig. 1. The Operation class, in purple, was added to the ontology as the mapping between the characteristics of R scripts and the existing classes in the ontology was performed. In addition to the classes described above, indicated in yellow, we have included in the diagram how

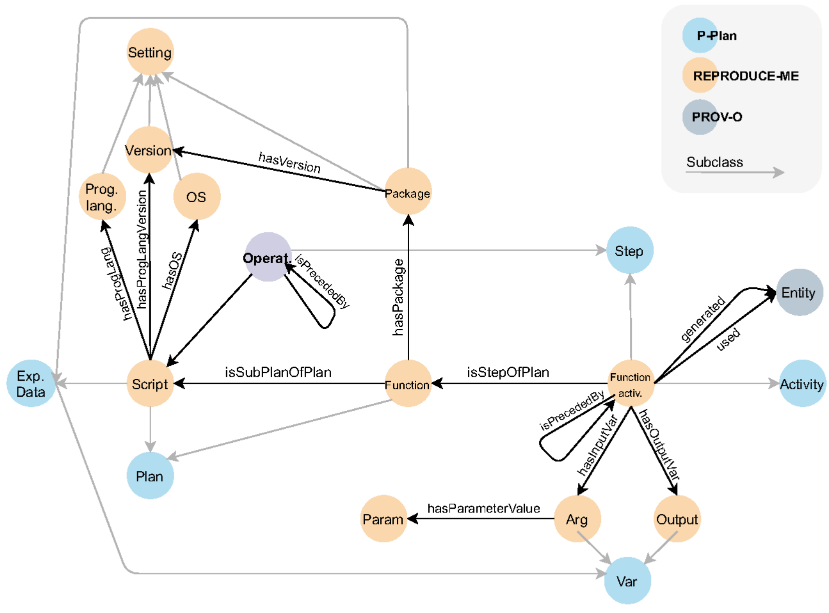

Fig. 1. Provenance representation of R scripts

they relate to some of the classes from the REPRODUCE-ME foundation ontologies (PROV and P-Plan). Details about these and other classes can be found in the ontology documentation[1].

The use of this vocabulary to represent provenance, as opposed to the conceptual model presented in our previous work, brings a set of benefits. REPRODUCE-ME, in its current state, details further information that may be useful to support reproducibility. It is a vocabulary based on other well-established models, allowing for better interoperability and suitability to the aspects involved in computational experiments. In addition, there is room for vocabulary extension to add other classes and relations as needed, which is quite useful considering this is a work in progress.

B. *Implementation of the provenance collection mechanism*

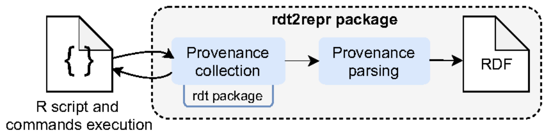

Fig. 2. Provenance collection process

As part of implementing a provenance collection mechanism and semantic representation according to the REPRODUCE-ME data model, we evaluated existing provenance packages that could aid this process. Among the options, we chose to implement this mechanism as an extension to the RDataTracker (or *rdt*) package[2]. This is because the package collects and records provenance data according to the PROV data model, which, as mentioned, composes the basis of the REPRODUCE-ME ontology.

In *rdt*, the PROV classes used include: *agent*, *entity*, *activity*, and *collections*; and relations between these classes include: *wasInformedBy*, *wasGeneratedBy*, *used*, and *hadMember*. *rdt* also extends the PROV model to include other properties used to represent provenance. The properties increase the detail level of the provenance, differentiating, for example, what are consumed and produced data entities from what are packages/libraries used during the execution of activities. In general terms, the provenance structure recorded by *rdt* has the following format: each R command executed, separately in the console or inside an R script, is an *activity*. The *wasInformedBy* relationship defines the flow of execution between activities. Input and output data or variables are *entities*. The *used* relationship connects input data with activities. On the other hand, the *wasGeneratedBy* relationship connects data and the activity that generated it. Packages, or libraries, are also *entities*. The *hadMember* relation connects activities that are functions and the package they belong to.

REPRODUCE-ME allows a greater detail level than the provenance recorded by the *rdt* package. To adapt the provenance of *rdt* to the semantic model, as presented in the previous subsection, we consider both the output produced by *rdt*, a JSON file containing the provenance, and the executed R script. Thus, for each class and relation expressed by *rdt*, we perform a process of checking with the script to match the REPRODUCE-ME classes and the appropriate relations. The correspondence of the *rdt* to the REPRODUCE-ME happens as follows: the activities are broken down into *function activations* or *operations*. For *function activations*, we record which *function* was executed and which *package* it belongs to. Unlike *rdt*, we also register the base R functions and not

[1]https://w3id.org/reproduceme
[2]https://github.com/End-to-end-provenance/RDataTracker

only the functions of other packages loaded in the script. For each *function activation*, we check the informed *arguments* and also if there are *arguments* with default values, which are not necessarily explicit in the function call. In this way, we can track the *parameters*' configuration every time a function is executed. Entities are broken down into *variables* if they comprise input and output data from *function activations* or *operations*, or just *outputs* if they are only generated as a result of *function activations* or *operations*.

The process of extracting the provenance collected by *rdt*, complementing it with reproducibility-related information, and representing it according to REPRODUCE-ME was implemented using R as a post-processing step to the provenance recording by *rdt*. For ease of use, we have made this mechanism available as an R package, named *rdt2repr*. We encapsulated *rdt* features into this package and added the option to do the extraction by calling a function. For users using RStudio[3], an integrated development environment (IDE) for R, this extraction function can also be executed from an *Add-in*, an option in the interface menu that acts as an extension to the IDE. The Fig. 2 shows how the steps for collection and representation of provenance according to REPRODUCE-ME occur. The user who wants to collect the provenance of R commands must indicate when the collection should start and end and then extract to REPRODUCE-ME. The result of this extraction is a file in RDF format. We envision improving this process to allow the inclusion of provenance into a database in an integrated way, facilitating, for example, the access, sharing, and executions of queries to retrieve relevant information.

C. Evaluation

As part of the demonstration and evaluation of the semantic representation of R scripts using the REPRODUCE-ME ontology, we considered a sample of an R script for collecting the provenance of its execution. This script uses a package to query tweets data from the Twitter API and execute a set of functions to analyze and save the obtained data in a file.

Once the script is defined, we load the implemented package, *rdt2repr*, and call the function that starts gathering the provenance of all commands. Then we run the script, and at the end, we call another function to finish the provenance collection. These two functions are native to *rdt* and have been encapsulated within the *rdt2repr* package. After that, we convert the collected provenance using the function implemented in *rdt2repr*. We define and select competency questions to answer based on the initial requirements of the computational reproducibility of scripts [5]. We demonstrate one of the questions here based on the generated RDF file, and the others, along with the R script used and the provenance collected, are available on GitHub[4]. As the provenance data is extracted to RDF format, we use SPARQL to define the queries. The following query lists the functions used in the definition of the sample R script, with the respective packages they belong to and the package versions used, the first lines of the result is presented in Table I:

[3]https://www.rstudio.com/

[4]https://github.com/mmondelli/rewords_sample

```
PREFIX repr: <https://w3id.org/reproduceme#>
SELECT DISTINCT *
WHERE {
  ?function a repr:Function ;
  repr:hasPackage ?package .
  ?package repr:hasPackageVersion
    ?package_version .
}
```

TABLE I
SAMPLE OF QUERY RESULTS

function	package	package_version
repr:pipe	repr:dplyr	”1.0.5”
repr:arrange	repr:dplyr	”1.0.5”
repr:count	repr:dplyr	”1.0.5”
repr:fwrite	repr:data_table	”1.14.0”
repr:library	repr:base	”4.0.2”

As an additional evaluation, we plan to work on the integration of the package developed in the framework proposed in the previous work, as well as identifying possible extensions in the ontology. In addition, we have two case studies of workflows developed in R by partner researchers, in ecological niche modeling and metabolome annotation, where we will be able to evaluate the work in greater depth.

IV. Conclusion

In the present work, we explore using a semantic approach to represent the provenance of R scripts through the REPRODUCE-ME ontology. We understand that adopting a semantic model that allows a greater description of the aspects involved in workflows defined through scripting languages can provide greater understandability and interpretability, and consequently reproducibility. Furthermore, as it is a model that extends well-consolidated ontologies, the REPRODUCE-ME brings the aspect of interoperability, which is essential if we consider supporting the other layers involved in computational experiments, also important to be represented from the point of view of reproducibility.

The present work and the development of the proposed R package act as proof of concept of applying the ontology in this context. We see in the application and use of a vocabulary that is essentially geared towards reproducibility and based on well-established ontologies an opportunity to support workflows following FAIR principles. With the approach proposed in this work, we can meet three of the FAIR guidelines, for the description of workflow with rich metadata (F2) using a formal vocabulary that can be widely applied to the context of computational experiments (I1 and I2). In this sense, we envision the integration of this work with the framework that we proposed previously to cover not only the representation of scripts but also other layers, such as the computational environment used to execute computational experiments. We intend to cover other guidelines such as assigning unique identifiers to the research objects involved in the experiment and supporting the reuse of the experiment and research objects (A1, F1, F2, and R1). This will demand extension of the ontology to include other classes and relations that may be necessary. Furthermore, we see the need to implement a mechanism that facilitates the consultation and visualization of source data that may be valuable to users to support the usability of the collected data.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] V. Stodden, D. Donoho, S. Fome, M. P. Friedlander, M. Gerstein, R. LeVeque, I. Mitchell, L. L. Oullette, and C. Wiggins, "Reproducible Research," *Computing in Science & Engineering*, vol. 12, no. 5, pp. 8–13, 9 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5562471/

[2] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, 5 2016. [Online]. Available: http://www.nature.com/doifinder/10.1038/533452a

[3] S. Samuel and B. König-Ries, "Understanding experiments and research practices for reproducibility: an exploratory study," *PeerJ*, vol. 9, p. e11140, Apr. 2021. [Online]. Available: https://doi.org/10.7717/peerj.11140

[4] P. Ivie and D. Thain, "Reproducibility in scientific computing," *ACM Computing Surveys*, vol. 51, no. 3, 2018.

[5] S. Samuel, "A provenance-based semantic approach to support understandability, reproducibility, and reuse of scientific experiments," Ph.D. dissertation, University of Jena, Germany, 2019. [Online]. Available: https://www.db-thueringen.de/receive/dbt_mods_00040396

[6] M. L. Mondelli, A. Townsend Peterson, and L. M. R. Gadelha, "Exploring reproducibility and fair principles in data science using ecological niche modeling as a case study," in *Advances in Conceptual Modeling*, G. Guizzardi, F. Gailly, and R. Suzana Pitangueira Maciel, Eds. Cham: Springer International Publishing, 2019, pp. 23–33.

[7] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten Simple Rules for Reproducible Computational Research," *PLoS Computational Biology*, vol. 9, no. 10, p. e1003285, 2013. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.1003285

[8] J. Kitzes, D. Turek, and F. Deniz, *The practice of reproducible research : case studies and lessons from the data-intensive sciences*, 2017. [Online]. Available: https://www.practicereproducibleresearch.org/

[9] S. Han, T. F. Olonisakin, J. P. Pribis, J. Zupetic, J. H. Yoon, K. M. Holleran, K. Jeong, N. Shaikh, D. M. Rubio, and J. S. Lee, "A checklist is associated with increased quality of reporting preclinical biomedical research: A systematic review," *PLOS ONE*, vol. 12, no. 9, p. e0183591, 9 2017. [Online]. Available: http://dx.plos.org/10.1371/journal.pone.0183591

[10] X. Feng, D. S. Park, C. Walker, A. T. Peterson, C. Merow, and M. Papeş, "A checklist for maximizing reproducibility of ecological niche models," *Nature Ecology & Evolution*, vol. 3, no. 10, pp. 1382–1395, 9 2019. [Online]. Available: http://www.nature.com/articles/s41559-019-0972-5

[11] Nature, "Checklists work to improve science," *Nature*, vol. 556, no. 7701, pp. 273–274, 4 2018. [Online]. Available: http://www.nature.com/articles/d41586-018-04590-7

[12] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X08000861

[13] J. Freire, D. Koop, E. Santos, and C. T. Silva, "Provenance for computational tasks: A survey," *Computing in Science and Engineering*, vol. 10, no. 3, pp. 11–21, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4488060

[14] F. Chirigati, R. Rampin, D. Shasha, and J. Freire, "ReproZip: Computational Reproducibility With Ease," in *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*. New York, New York, USA: ACM Press, 2016, pp. 2085–2088. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2882903.2899401

[15] T. Pasquier, M. K. Lau, X. Han, E. Fong, B. S. Lerner, E. R. Boose, M. Crosas, A. M. Ellison, and M. Seltzer, "Sharing and preserving computational analyses for posterity with encapsulator," *Computing in Science and Engineering*, vol. 20, no. 4, pp. 111–124, 2018.

[16] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "noWorkflow: a Tool for Collecting, Analyzing, and Managing Provenance from Python Scripts," in *Proceedings of the VLDB Endowment, Vol. 10, No. 12*, 2017, pp. 1841–1844. [Online]. Available: http://www.vldb.org/pvldb/vol10/p1841-pimentel.pdf

[17] W. M. Landau, "The drake r package: a pipeline toolkit for reproducibility and high-performance computing," *Journal of Open Source Software*, vol. 3, no. 21, 2018. [Online]. Available: https://doi.org/10.21105/joss.00550

[18] B. Lerner, E. Boose, L. Perez, B. Lerner, E. Boose, and L. Perez, "Using Introspection to Collect Provenance in R," *Informatics*, vol. 5, no. 1, p. 12, 3 2018. [Online]. Available: http://www.mdpi.com/2227-9709/5/1/12

[19] S. Samuel and B. König-Ries, "ProvBook: Provenance-based Semantic Enrichment of Interactive Notebooks for Reproducibility," in *17th International Semantic Web Conference (ISWC) 2018 Demo Track*, 2018. [Online]. Available: http://ceur-ws.org/Vol-2180/paper-57.pdf

[20] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *ACM SIGMOD Record*, vol. 34, no. 3, p. 31, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1084805.1084812

[21] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche, "The Open Provenance Model core specification (v1.1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, 6 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X10001275

[22] D. Garijo and Y. Gil, "Augmenting PROV with Plans in P-PLAN: Scientific Processes as Linked Data," in *Proceedings of the Second International Workshop on Linked Science*, Boston, MA, USA, 11 2012. [Online]. Available: http://ceur-ws.org/Vol-951/

[23] P. Missier, S. Dey, K. Belhajjame, V. Cuevas-Vicenttin, and B. Ludäscher, "D-PROV: Extending the PROV Provenance Model with Workflow Structure," in *5th Workshop on the Theory and Practice of Provenance, (TaPP'13)*. Lombard, IL: USENIX Association, 4 2013. [Online]. Available: http://www.dataone.org

[24] V. Cuevas-Vicenttín, B. Ludäscher, P. Missier, K. Belhajjame, F. Chirigati, Y. Wei, S. Dey, P. Kianmajd, D. Koop, S. Bowers *et al.*, "Provone: A prov extension data model for scientific workflow provenance," 2014. [Online]. Available: https://purl.dataone.org/provone-v1-dev

[25] F. Costa, V. Silva, D. De Oliveira, K. Ocaña, E. Ogasawara, J. Dias, and M. Mattoso, "Capturing and Querying Workflow Runtime Provenance with PROV: a Practical Approach," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops on - EDBT '13*. New York, New York, USA: ACM Press, 2013.

[26] M. D. Wilkinson *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific data*, vol. 3, 2016.