

OPEN SCIENCE

SOURCE CODE AND SOFTWARE



PASSPORT FOR
OPEN
SCIENCE

What is software?	4
Why open up software?	6
How to open up software?	7
What's next? Preparing for the future	10
Going further	12
Sources	13
Glossary	14

Key

Underlined text is explained in the glossary.

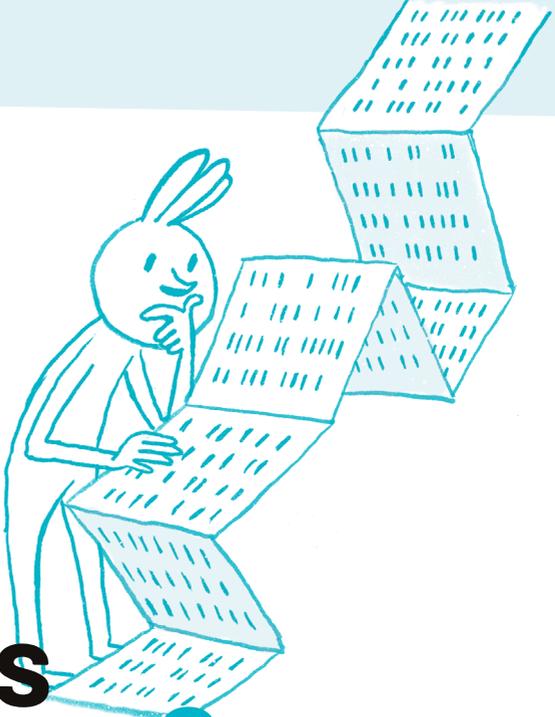
▼ refers to tools which are given as examples.

The digital version of this guide is available at www.ouvri.lascience.fr

As part of the Passport for Open Science collection, this booklet addresses the particular challenges of opening up the source code and software produced and used in scientific research.

What legal status do source code and software have? Why and how should we make them open access? How best to structure their development and ensure permanent archiving? How should we approach promotion? This guide provides answers and tools that you will immediately be able to put into practice. Here you will find the keys to understanding the role of software in the open science ecosystem. We hope this will encourage you to make a lasting contribution to the work of the academic community.





what is software?

The ins and outs of software

Software (or a computer program) is the description, in one or more programming languages, of the way you want to get a computer to process data. **The source code** is the basic element of any software. It is the text written in programming languages by one or more human authors. There are many different languages, each corresponding to specific needs: C, Java, Python, R, Ocaml, Scilab, etc. Computers are only capable of executing elementary instructions, written in a low-level machine language. To execute source code on a computer, you therefore need to:

- either **compile** it, which means converting it beforehand into a functionally equivalent object code written in machine language that can be executed on the computer;
- or interpret it, which means translating it on the fly within a dedicated environment (a Python interpreter, etc.). Small source codes intended to be interpreted are often called scripts. Tiny scripts are sometimes called macro-instructions, or macros.

GOOD TO KNOW

Everyone creates code, sometimes without realising it! Even the calculation formulas in a spreadsheet are source code: they are small scripts written in the software's macro language, which are meant to be interpreted each time the user changes the value in a cell. Graphic representations of calculations in virtual lab software are also code, which can be expressed in text form.

We use the term software when the code is useful enough in itself to be considered to exist in its own right in terms of its preservation, evolution, and distribution. In this case, the software might exist in several versions, which either coexist or follow one another over time, as the initial authors or other contributors make adaptations.

Some software can be used independently, while others are designed to provide a specific service and are embedded in other programs as software components.

Also, a computation process can be described in a notebook which combines fragments of software with descriptive information (documentation or explanation).

Legal status

Software is considered as an intellectual creation and as such is protected by authors' rights. There are two facets to this:

- economic rights, which relate to exploitation of the work. These rights are exercised by the rights holders, i.e. natural or legal persons who may be different from the author. For example, they may be the author's employer, if the author is an employee or civil servant;
- moral rights, which exist in the European Union and in other legal systems, which relate to protection of authorship. Even if the work is exploited by rights holders, authorship will always remain with the authors, i.e. the natural persons who wrote the source code.

When a computer program is made operational or is distributed, a license may be associated with it. This agreement sets out the rights and duties of those who receive or use the software. **The license defines the degree of openness of the code.**

Anything that is not explicitly authorised by the license is forbidden. If a software program's source code is distributed without a license, only consultation of the code is authorised.

Software often comes with documentation, which is also protected by author's rights. It might be:

- internal to the code, in the form of comments;
- external to the code, in the form of maintenance or user manuals.

why open up software?



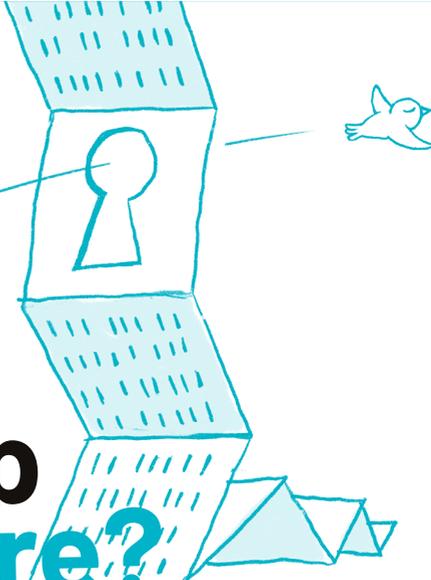
Many scientific results rely on computerised data processing. Their **reproducibility** depends on the scientific community being given the means to perform anew the experiments using **software**. Yet it is almost impossible to guarantee the identical execution of software for all users over time. Opening up software is one way to reduce this difficulty. It is an approach that must be combined with opening up research data. Check out the **▼MOOC** on Reproducible research to learn more.

Many initiatives now facilitate the integration of software to publications, including **▼IPOL**, for image processing, **▼eLife**, for the life sciences, or the **▼Graphics Replicability Stamp Initiative**.

Because it makes it easier to compare results, **opening up software boosts peer citations**. It encourages collaboration on the created software, by expanding the **community** of users or **contributors**. These communities can be federated thanks to collaborative platforms, such as the **▼ropensci** initiative for the programming language R.

Opening up codes and software also helps **improve** them and fosters the **emergence of new ideas**. It stimulates scientific research in a spirit of cooperation (cooperation and competition). Providing access to the software is a way to **promote the skills** of its authors and their teams, and to give them additional recognition.

Finally, opening up software is an integral part of national and European policies on open science, as in the case of **French Law for a Digital Republic** (2016) and the **French second National Plan for Open Science** (2021).

An illustration featuring a stack of four papers or folders. The top paper has a keyhole cutout. A bird is flying to the right, with a line connecting it to the keyhole. Below the stack is a small pyramid. The text 'how to open up software?' is overlaid on the left side of the illustration.

how to open up software?

If you would like to share your software and allow others to use it, you must organise how it is accessed both legally and technically. You also need to think about the future of its source code in the long term: pursuing research activities related to it and/or implementing an active promotion approach.

Choosing one or more licenses

Choosing the license is an important step in the open access process, and one that can have significant consequences for the software's evolution.

The first step is to identify who holds the economic rights on the software: if the authors are not the rights holders, it is not up to them to choose the license.

The person, group or entity responsible for this choice can adopt a completely open approach and use a free software license. The ▼**Etalab** department of ▼**DINUM** provides a ▼**list of licenses that can be used for productions of French public authorities**. In other cases, you may wish to refer to the lists compiled by the ▼**Open Source Initiative** or ▼**Free Software Foundation**. In some situations, it is possible to use more restrictive licenses that only allow the code to be consulted or limit use by third parties. For instance, usage can be limited to an academic context for scientific reproducibility only.

If the software uses pre-existing components, you must ensure:

- that the license chosen is legally compatible with those of the different pre-existing components;
- that the licenses of the different components are compatible with one another.

PRATICAL EXAMPLES

There are different types of licenses, each one adapted to different practices in terms of reuse. **Scikit-learn** is a state-of-the-art statistical learning library that makes the field accessible to all. Designed to be embedded in other software, this library is distributed with a permissive license, the BSD. **WebObs** is a multidisciplinary real-time observation tool used to study natural phenomena. It has been adopted by several countries (Indonesia, Singapore, Peru) to monitor their volcanoes and is distributed under a diffusive license, the GPL 3.0. **Scikit-learn** and **WebObs** are two examples of software that won the Open Science Awards for Research Software (2022).

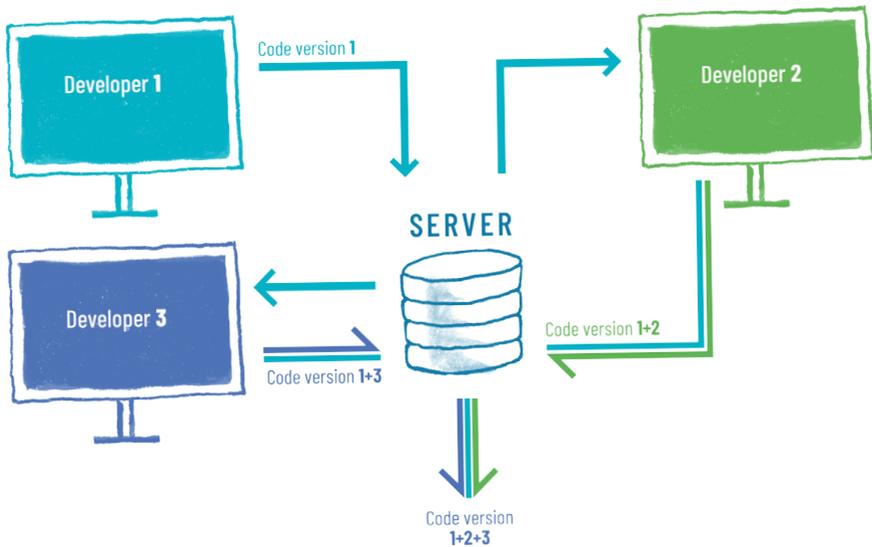
Organising software's development

Maintaining, organising and developing software of a certain size requires adopting good practices. Evolutions in the source code must be traced using a version control system: ▼**Git**, ▼**Mercurial** or ▼**Subversion**.

Many web platforms, known as software forges (▼**bitbucket.org**, ▼**gitlab**, ▼**github**, etc.), simplify these tasks and facilitate the aggregation of communities of contributors. Local instances of these platforms can be put in place by the home institution or by a research infrastructure, such as ▼**HumaNum** for the humanities and social sciences.

Each platform's communities can provide useful benefits: looking for bugs, adding new features, ensuring the software's long-term sustainability, managing contributors and their permissions, implementing quality assurance tests, etc. To benefit from the contributions of a community, one must invest the time and energy needed to organise its functioning: rapidly evaluate and deal with suggestions and contributions made, and plan the roadmap for the software's development.





GOOD TO KNOW

Software forges are not permanent archives: the projects they contain can be modified or deleted. Entire forges can also disappear, as happened to Google Code and Gitorious in 2015, and part of BitBucket in 2020. It is therefore preferable to choose a forge that is managed by a public body and recommended by your institution, and also to ensure that your codes are properly archived over time in dedicated software archives like **▼Software Heritage**.

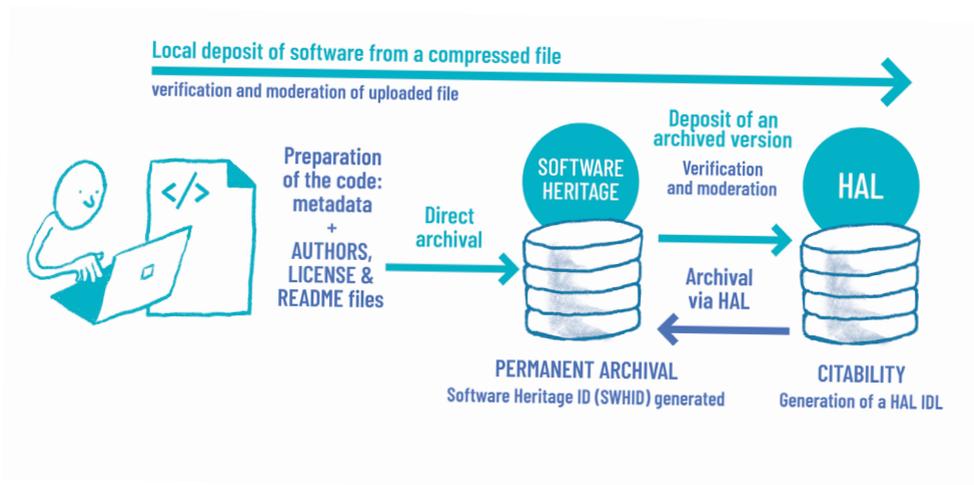
preparing for the future



Archiving, referencing and describing software

As with publications and data, it is important to ensure the long-term preservation of research software by archiving it on a suitable platform. The versions used must be referenced with precision, otherwise it will be hard to reproduce the results obtained. It is important to take the time to describe the software, using README and AUTHORS files and adding metadata for easier indexing.

The Software Heritage platform provides a very easy way to archive and reference embargo-free and publicly available software. It is also possible to deposit directly in **▼HAL**. This open archive can be used to manage the description and citation of research software, in collaboration with **▼Software Heritage**.



Software promotion

Making the source code open is not enough. In order for your code or software to be used, some communication actions are necessary: posting messages on distribution lists, social media or institutional websites. You must provide a brief explanation of why you created the tool in the first place. For more technical specifications, point users towards the documentation you designed. After the launch phase, communication must be managed in the long term: **if you are running a training course or demonstration, publishing something in relation to your software, or releasing a new version, let people know!**

When a piece of software becomes strategic for members of its community, they may decide to opt for a more formal organisational structure to ensure the software's long-term sustainability and development. Promotion efforts could for example lead to the creation of a club of partners (or consortium), funded by its members, or even a dedicated company.

Depending on the policy defined collectively by rights holders, the same software can be made available under different licenses for different user groups. A community version could be made available under an open source license, whereas a commercial version may be offered to industrial partners, perhaps with a different license more suited to their needs, in the context of paid contracts. To facilitate the management of software rights, a mandate of valorisation can be given to one rights holder by the others, to negotiate in their name. It is also possible over time to arrange for the transfer of contributors' rights holderhood to a single right holder, such as a dedicated foundation, better positioned to represent their interests.

IN THE FIELD

Arnadi Murtiyoso, a recent PhD graduate from the University of Strasbourg working in photogrammetry and geomatics, currently a post-doc at the ETH in Zurich

As part of my thesis, we developed ARCh, a benchmark for 3D heritage documentation. We used a software forge and placed our code under a GPL 3.0 license, which enables other researchers to reuse it to develop a benchmark in their own field of study. Opening up research software is essential for the understanding and analysis of results, as well as for the control of the processes involved.

Citing software

When writing a scientific article, it is best to give credit to the authors of the software used by adding the proper citation. There are now bibliographic styles dedicated to software that are gradually being adopted. You can also indicate the codes and software you produced in your **ORCID** account.

The **biblatex-software** package available on **CTAN** is capable of managing bibliographical entries for software. These can also be imported into **Zotero** using the **BetterBibLatex** extension.

Going further

ARCHIVAL, REFERENCING, DESCRIBING AND CITING

A guide on how to archive and reference source codes in ▼ **Software Heritage**.

<https://www.softwareheritage.org/howto-archive-and-reference-your-code/>

Guide on depositing software with metadata files in ▼ **HAL**.

<https://doc.archives-ouvertes.fr/en/deposit-2/deposit-software-source-code/>

An overview of how to cite software.

<https://www.softwareheritage.org/2020/05/26/citing-software-with-style/?lang=fr>

LEGAL ASPECTS

▼ List of licenses that can be used for dissemination by French public bodies.

<https://www.data.gouv.fr/fr/pages/legal/licences/>

List of licenses with relevant comments by the ▼ **Free Software Fondation**.

<https://www.gnu.org/licenses/license-list.en.html>

Text by the World Intellectual Property Organization on copyright protection of computer software.

<https://www.wipo.int/copyright/en/activities/software.html>

P. Moreau, C. Moulin, J. Pappalardo, F. Pellegrini (2017). Fondamentaux juridiques.

Collaboration et innovation ouverte, 2nd edition, *Les livrets bleus du logiciel libre*.

https://cnll.fr/media/LivretBleu_Juridique-2eEdition_GT-LogicielLibre_Systematic_Nov2016_web.pdf

RESEARCH SOFTWARE NETWORKS

French network for reproducible research. <https://www.recherche-reproductible.fr/>

French network of contributors to software development in higher education and research.

<https://www.devlog.cnrs.fr/>

COMPUTER SCIENCE (GENERIC)

V. Doutaut (2021). Informatique et culture scientifique du numérique, une transcription des

MOOC réalisés par le Learning Lab Inria. <https://hal.inria.fr/hal-03346079>

METADATA

Platforms to verify code and generate metadata:

▼ **CodeMeta Projet** (<https://codemeta.github.io/>) et

▼ **CodeMeta generator** (<https://codemeta.github.io/codemeta-generator/>)

CONTRIBUTIONS

▼ **CRedit**, which provides a description of the different roles of contributors to scientific output. <https://credit.niso.org/>

VERSION CONTROL SYSTEMS

Git, a website offering resources related to Git. <https://git-scm.com>

▼ **Learn Git Branching**, an application to help you understand the concepts behind the

branches when working with Git. https://learngitbranching.js.org/?locale=fr_FR

VERIFYING AND PUBLISHING CODE

▼ **CODECHECK**, an IT platform that can be used to verify the underlying software in scientific articles. <https://codecheck.org.uk>

CAdvice on publishing software. GitHub, Tips for Publishing Research Code.

<https://github.com/paperswithcode/releasing-research-code/>

Sources

- P. Alliez, R. Di Cosmo, B. Guedj, A. Girault, M. S. Hacid, A. Legrand, N. Rougier (2019).** Attributing and Referencing (Research) Software: Best practices and outlook from Inria. *Computing in Science and Engineering, Institute of Electrical and Electronics Engineers*, pp.1-14. <https://hal.archives-ouvertes.fr/hal-02135891v2>
- Article L. 113-9 of the French Intellectual property code.** https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000039279818/
- L. A. Barba.** Terminologies for reproducible research. arXiv:1802.03311 [cs.DL]. <https://doi.org/10.48550/arXiv.1802.03311>
- G. Colavizza, I. Hrynaskiewicz, I. Staden, K. Whitaker, B. McGillivray (2020).** The citation advantage of linking publications to research data. *PLOS ONE* 15(4): e0230416. <https://doi.org/10.1371/journal.pone.0230416>
- L. Courtès (2021).** Reproduire les environnements logiciels: un maillon incontournable de la recherche reproductible. *Bulletin de la Société informatique de France*, n° 18, pp. 15-22. <https://dx.doi.org/10.48556/SIF.1024.18.15>
- R. Di Cosmo.** biblatex-software – BibLATEX stylefiles for software products. <https://ctan.org/pkg/biblatex-software>
- L. Desquilbet, S. Granger, B. Hejblum, et al.** Vers une recherche reproductible : faire évoluer ses pratiques. <https://hal.archives-ouvertes.fr/hal-02144142>. Available as a collaborative version at <https://rr-france.github.io/bookrr/>
- French decree n° 2017-638 of 27 April 2017** on licenses for the free reuse of public information and the procedures for their approval. <https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000034502557>
- M. Gruenpeter, J. Sadowska, E. Nivault, A. Monteil (2022).** Create software deposit in HAL: User guide and best practices. [Technical Report] Inria; CCSD; Software Heritage. <https://hal.inria.fr/hal-01872189>
- J. Kitzes, D. Turek, et F. Deniz (2018).** *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences*. Oakland: University of California Press. www.practicereproducibleresearch.org/
- Open Science award for free research software.** <https://www.enseignementsup-recherche.gouv.fr/fr/remise-des-prix-science-ouverte-du-logiciel-libre-de-la-recherche-83576>
- J.-F. Pimentel, L. Murta, V. Braganholo, et J. Freire (2021).** Understanding and improving the quality and reproducibility of Jupyter notebooks, *Empirical Software Engineering*, 26, 65. <https://doi.org/10.1007/s10664-021-09961-9>
- H. E. Plesser (2018).** Reproducibility vs. Replicability: A Brief History of a Confused Terminology, *Frontiers in Neuroinformatics*, 11:76. <https://doi.org/10.3389/fninf.2017.00076>
- K. Powell (2020).** Tech tools to make research more open and inclusive. *Nature*, 578, 181-182. <https://doi.org/10.1038/d41586-020-00216-z>
- G.K.Sandve, A. Nekrutenko, J. Taylor, E. Hovig (2013).** Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology* 9(10): e1003285. <https://doi.org/10.1371/journal.pcbi.1003285>
- Utrecht University (n.d.).** Best Practices for Writing Reproducible Code. Workshop-Computational-Reproducibility. <https://utrechtuniversity.github.io/workshop-computational-reproducibility/>

Glossary

Archiving: ensuring the lasting conservation, access to and indexing of a particular version of a piece of software. Because it is not sufficient to make software available on a professional or laboratory website to achieve these goals, it is best to use a dedicated platform.

Authors' rights: legal regimes governing the exploitation of intellectual creations. These rights may differ depending on the kind of intellectual creation, e.g.: software code, preparatory design material, documentation, graphics and sounds of the user interface.

Club of partners/Consortium: coalition with a shared strategic interest in promoting the development and/or maintenance of a particular software or standard.

Community: all those who use or are likely to contribute to a software project.

Compiling: translating source code into object code, generally to obtain a machine code to be executed on a particular type of computer. The compiler is the software used to complete this translation automatically.

Computational notebook: document that can contain executable code, text, mathematical formulas, graphs and interactive media (e.g. a [▼ Jupyter Notebook](#)).

Contributor: someone who takes part in the development of a software project, whether by producing source code or engaging in other activities: testing, translating the interface or documentation, taking an active role in the community, etc.

Executable code: code written in machine language, that can be directly executed on a particular type of computer. It is usually an object code.

Forge: environment for software development that facilitates collaborative work on a software project. A forge contains tools such as a versioned source code repository, discussion forums, an automated test environment, etc.

Intellectual creation: creation of form suitable for protection by copyright as it expresses its author's personality (known as the "originality" criterion).

Interface: set of conventions for using a software module.

Legal compatibility: ability of two licenses covering two distinct models to coexist within the same software. For example, a module under the GNU GPL free software license and another module under a non-free license cannot be incorporated into the same software, unless in exceptional circumstances.

License: contractual document by which the rights holder grants certain permissions to users. A license is said to be "free" if it grants all of the following four freedoms: usage, copies, modification and redistribution of the modified software.

Library (or Software component or Module): software providing a specific service and intended to be incorporated into another software.

Machine language: rudimentary, low-level language used to write code that can be executed directly on a specific type of computer. Programmers prefer to write their source codes in higher-level languages which are more expressive. The source codes must then be translated into object code that can be executed on the computer, usually by compilation.

Macro: tiny script.

Mandate (of valorisation): agreement between joint rights holders to appoint one of them to carry out administrative procedures on their behalf.

Metadata: data associated with other data which they contextualise (e.g. timestamp and geolocation of a photo).

Object code: code not directly written by a human author. It is the result of the translation of source code using a tool such as a compiler.

Programming language: language specifically designed to describe computation processes likely to be performed by a computer.

Referencing: unequivocal identification of objects. These can have different levels of granularity: a complete version of a software program, a file, a block of lines of code. Referencing requires the attribution of unique and permanent IDs like the SoftWare Heritage persistent Identifier (SWHID).

Reproducibility: ability to reproduce the results of a study by replicating the original process.

Script: fragment of code that will be interpreted in a work environment such as office software, a command line interpreter, etc.

Software: text, written in one or more programming languages, describing the calculations to be performed by a computer.

Source code: code written by a human author.

Version: copy of a software program made available at a particular date to a given population of users. A given software version can be associated with a specific license or specific services (user assistance, bugfix support, etc.).

Credits

Direction of publication

Ministry of Higher education and Research

Editorial coordination

University of Lille

Scientific council

The Skills and Training and Codes and Software Colleges of The Committee for Open Science

Project leader

Mónica Michel Rodriguez

Writers

François Pellegrini, Roberto Di Cosmo, Laurent Romary, Sabrina Granger, Sacha Hodencq, Joanna Janik, Romane Coutanson, Madeleine G eroudet

Translator

Proofread by : Fran ois Pellegrini, Roberto di Cosmo, Jennifer Morival, Connor Alexander Mayon

Graphic design

Studio 4 minutes 34
Studio Lendroit.com

Printing

L'Art sienne, Li vin

1st edition: August 2022

Thanks

Arnadi Murtiyoso, who shared his experience of open science

Early-career researchers who provided feedback on the first version

Julien Cicero, Erwin Gerard, Aldo Gonzalez Lorenzo, Lo ck Kl eparski, C dric Marinel, Jean-Paul Martischang, Martin Mion-Mouton, Antoine Olczak

Experts consulted

Isabelle Blanc, Danielle Bourcier, Franck Macrez, Sofia Papastamkou

This guide is part of the Passport for Open Science collection.

The digital version of this guide is available at www.ouvri.lascience.fr

This guide is made available under the terms of the *Creative Commons* CC BY-SA 4.0 license. Attribution - Sharing subject to the same terms.

