Open Science in Software Engineering

Daniel Méndez Fernández, Daniel Graziotin, Stefan Wagner, and Heidi Seibold

Abstract Open science describes the movement of making any research artefact available to the public and includes, but is not limited to, open access, open data, or open source. While open science is becoming generally accepted as a norm in other scientific disciplines, in software engineering, we are still struggling in adapting open science to the particularities of our discipline, rendering progress in our scientific community cumbersome. In this chapter, we will reflect upon the essentials of open science for software engineering including what open science is, why we should engage in it, and how we should do it. We particularly draw from our experiences in chairing roles implementing open science initiatives and researchers engaging in open science to critically discuss challenges and pitfalls, and to address more advanced topics such as how and under which conditions to share preprints, what infrastructure and licence model to cover, or how do it within the limitations of different reviewing models, such as double-blind reviewing. Our hope is to help establishing a common ground and to contribute to make open science a norm also in software engineering.

Daniel Mendez
Technical University of Munich, Boltzmannstraße 3, 85748 Garching, Germany. e-mail: daniel.mendez@tum.de

Daniel Graziotin
University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany. e-mail: daniel.graziotin@iste.uni-stuttgart.de

Stefan Wagner
University of Stuttgart, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany. e-mail: stefan.wagner@iste.uni-stuttgart.de

Heidi Seibold
Ludwig-Maximilians-University Munich, Marchioninistraße 15, 81377 Munich, Germany. e-mail: hseibold@ibe.med.uni-muenchen.de

1 Introduction

In a nutshell, open science refers to the movement of making any research artefact available to the public. This ranges from the disclosure of software ("open source"), over the actual data itself ("open data") and the material used to analyse the data (such as analysis scripts, "open material") to the manuscript reporting on the study results ("open access").

By disclosing research artefacts, we often do so in the expectation of increasing transparency and, thus, reproducibility and replicability of our scientific process and our results. The principles associated with open science are often seen as an important means to move forward as a scientific research community. Open data and open source – both being major principles under the common banner of open science – constitute a major hallmark in making empirical studies transparent and understandable to researchers not involved in carrying out those studies. This can be done, for example, by sharing replication packages that capture the raw data and anything necessary for their analysis and interpretation. That way, we increase the reproducibility of our research. This, in turn, strengthens the credibility of the conclusions we draw from the analysed data and it allows others to build their own work upon ours; hence, it strengthens more generally our overall body of knowledge in the community.

Besides these more ideological views on open science and reasonable arguments in favour of engaging into it as a research community, on which any reader will probably agree, there is much more to it which we need to understand when considering open science in the context of software engineering research. There are, for example, various challenges in data disclosure – technical, ethical and legal ones, but also social ones – which are different to the standards and views given in other disciplines and which make open science difficult to become the norm in our own field. Consider, for example, the notion of repeatability, replicability, and reproducibility by considering the terminology as introduced by the ACM[1] (verbatim):

- **Repeatability (Same team, same experimental setup):** The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.
- **Replicability (Different team, same experimental setup):** The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artefacts.
- **Reproducibility (Different team, different experimental setup):** The measurement can be obtained with stated precision by a different team, a different

[1] `https://www.acm.org/publications/policies/artifact-review-badging`

measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artefacts which they develop completely independently.

As an engineering discipline coming from the natural science, we often make implicit assumptions that our focus is on quantitative and even purely computational studies (e.g. simulations). For these, existing definitions and norms hold as they are and we are able to yield replicability and reproducibility. This situation is, however, not the norm. Many studies in Software Engineering involve – in one form or another – humans. In the end, software is made by humans for humans. Human subjects, however, act purely rational in exceptional cases only, if at all [20]. This means that every change of such an experimental context, even if strictly following the same experimental setup, will eventually yield different (context-dependent) results. Such studies would then not fit the available definition of reproducibility as used in computational studies, but it is still reasonable to argue that they would still be reproducible. Another challenge in software engineering research is that much of our data emerges from sensitive (e.g. industrial) settings. This renders full disclosure often difficult and we often need to anonymise the data to act within legal and ethical constraints that most computational studies do otherwise not have. Those two facets of software engineering research alone show already that we need to adopt open science principles to the particularities of our discipline, same as it is the case in other disciplines, too.

How can our software engineering community of researchers adopt its own open science movement? We believe that it is the lack of proper understanding about

- what open science is (and what it isn't) for software engineering,
- why we should all do our best to implement it, whether as editor, chair, or as researcher, and finally
- how we could and should do it

that often yields a general reluctance towards implementing open science and, sometimes, even a general dismissal of the potential open science has for individual researchers and the community as a whole. All this renders our own open science movement cumbersome.

In this book chapter, we will cover the essentials in open science for software engineering. In particular, we will establish a common ground in our discipline by elaborating on established key terms, principles, and approaches – all tailored to the particularities of our discipline. We will further discuss practical guides to implementing open science before concluding with a discussion of chosen challenges and pitfalls. The latter are based on our shared experiences emerging from open science activities and lessons we learnt so far as authors and as organisers where we implemented first open science initiatives in the empirical software engineering community.

The main addressees are software engineering scholars interested in the general notion of open science and those interested in implementing open science in their own research practices. One hope we associate with this manuscript is not only to oppose those critical voices still sceptical towards open science, but also to strengthen

the voices of those supporting it out of the firm conviction that open science should soon become the norm in software engineering research, too.

2 What is Open Science?

Open science is a movement whose scope is to render all artefacts born out of scientific research activities accessible, without any barriers, to any individual on Earth [35]. Open science refers also to the scientific part of the broader terms of open scholarship, i.e. "the process, communication, and re-use of research as practised in any scholarly research discipline, and its inclusion and role within wider society." [30]. Open science itself is an umbrella term that encompasses several facets of openness, for example open access, open data, open source, open government, open notebooks, or open standards [7]. In the following, we will discus those concepts particularly relevant to the (empirical) software engineering research community.

2.1 Open Access

Open access is associated with publications, i.e., research articles, technical reports and papers in general. Open access occurs whenever a publication is freely available on the public Internet without any access barrier – financial, legal or technical ones (including even not to force users to register to systems). It allows individuals to read, download, copy, distribute, print, search or link to the full texts of publications for any lawful purpose [4]. Minor constraints over redistribution and reuse of the publication may still apply and usually take the form of attribution. It is typical with open access publications that the authors retain the copyright of their work, and the act to render the work as open access is enabled through proper licenses. The *Creative Commons* licence model is the most widely employed one for open access.

Open access can take several forms according to which version of a publication is made public and at which point of the academic writing process it is made public. If authors make an own produced copy of their work openly available, they perform an act of *self-archiving*. The work is called *preprint* if it reflects a version of their manuscript that has not yet been accepted for publication in a scientific venue. If the content of the own produced work is identical to the content of the accepted publication, it is called *postprint*. The only differences between the *postprint* and the manuscript formally published by a traditional publisher like ACM, IEEE, or Springer lie in typesetting differences and the location of the document. The location of pre- and postprints is typically an open repository for pre- and postprints, in contrast to the digital libraries of the publishers. One such example is given in the following while we will go more into detail in Sect. 4.

Self-archiving via arXiv

arXiv, pronounced as *archive* and available at `https://arxiv.org`, is a repository, born in 1991, of freely accessible preprints and postprints, as well as whitepapers, covering several scientific fields including physics, mathematics, and computer science [12]. arXiv is free to access, register to, and submit to, but it presents two barriers for publishing. First, authors have to be endorsed by existing members before they are allowed to register in the system. Second, every submission is moderated by volunteers who check for issues such as scope or copyright. arXiv is the de-facto standard repository for mathematics and physics, and with some authors only publishing their work in there, it receives more than 10000 submissions per month and is, at the time of writing this chapter, hosting approximately 1.5M manuscripts in a distributed archived system of multiple digital libraries all over the world.

The act of self-archiving is also known as *green open access* and it is allowed by the majority of academic publishers with some regulations.

Checking self-archiving options in tune with publishers' regulations

Different publishers define different regulations with effect to the needs and possibilities of self-archiving, and it is imperative to strictly adhere to these rules. The so-called SHERPA partnership, a partnership of several universities with the original goal of setting up an institutional open access repository, offers with *RoMEO* – `http://www.sherpa.ac.uk/romeo` – a tool summarising publishers' copyright and archiving policies. RoMEO distinguishes different categories via the following colour codes commonly adopted also in the wider sense:

- **White:** Self-archiving not formally allowed
- **Yellow:** Authors can archive pre-prints (i.e. pre-refereeing)
- **Blue:** Authors can archive post-prints (i.e. final draft post-refereeing) or publisher's version/PDF
- **Green:** Authors can archive preprint and postprint or publisher's version

---

Whenever a publisher renders an accepted publication as openly licensed and available without any restriction whatsoever, the artefact becomes open access under the so-called *gold open access model*. We refer the reader to the work of Graziotin et al. [15] for more information on open access and its publishing models.

2.2 Open Data

Open data is very similar to open access, but it is applied to any data being produced in the course of research activities, such as the raw data obtained via a controlled experiment. Open data follows the idea that research data should be freely available to everyone to use and redistribute as they wish, without any restriction whatsoever

born out of copyright and licenses [3]. As with open access, the Creative Commons deeds are commonly employed licences for open data.

Creative Commons (CC) copyright licences

Creative Commons copyright licences (see `https://creativecommons.org/licenses/`) constitute a public licence model with the aim to facilitate granting copyright permissions to published work. The two most employed Creative Commons deeds are the Public Domain (CC0, "No rights reserved") and the Attribution 4.0 (CC BY 4.0) license. The former is a license that implements true public domain, effectively acting as a renounce of any copyright on the artefacts. The latter is an open license that allows reuse and redistribution of the artefact with the only condition of attributing the original work to the authors.

Besides the frequently used CC licence models introduced above, further ones are possible, too. One example is the Attribution-NonCommercial 4.0 (CC BY NC 4.0), which adds the clause that the original artefact and any derivation of it cannot be used for commercial purposes. While the Public Domain and the CC BY NC licenses might seem more suitable for academic work, opting for them can be problematic as we will explain in section 5.3.

2.3 Open Source

Open source in open science is nothing different to open source software as it is commonly known by the computer science community. Several research endeavours in computer science and empirical software engineering, but also other disciplines as well, produce software. One such example is what is often referred to as *research software*, i.e. software products developed with the purpose of analysing (empirical) data, such as Python code. In principle, the software developed can be released as open source software using known licenses such as the MIT license or the GPLv3.

2.4 Preregistration of Studies

Preregistration is a useful tool to ensure a certain level of quality of a study design, e.g. by making sure that hypotheses of a confirmatory study were actually predefined rather than being defined after having analysed the data to fit the results. Researchers define what their research questions are, why they want to pursue the research, and how exactly they will try to answer their questions. The Open Science Framework is currently one of the most common places to preregister research

projects (see https://osf.io/prereg/). Some journals are noticing already how preregistration avoids

- publication bias [9],
- p-hacking [16], and
- HARKing (Hypothesizing after the results are known [18]).

These journals offer the possibility of submitting a so-called *registered report* to their journal. For a guide on writing registered reports, we refer the reader to https://osf.io/8mpji/. Such a report goes through peer review and, provided acceptance, the report is *in principle accepted* (IPA). If the researchers conduct the study as indicated in the registered report, their paper will be published in the journal regardless of the results.

2.5 Open Science Badges

For every form of open science, publishers can award so-called *Open Science Badges*. Badging is a form of promoting open science activities of researchers via a specific badge that publicly recognises their open science engagement. To this end, publishers associate a specific symbol (i.e. a badge) to chosen artefacts to certify that the content is available and accessible in a persistent location.

There exist various forms of badges obeying the particularities of the various available badge systems. Some of them are publisher-specific (such as the ACM badge system[2]) and some of them are independent, such as the OSF Open Science Badges.

OSF Open Science Badges

A wide-spread open science badge system is the one introduced by the Open Science Foundation (OSF, https://osf.io/) and further promoted by the Center for Open Science (https://cos.io). This model distinguishes between badges in the following categories:

- **Open Data:** This badge is awarded when shareable data necessary to reproduce a study are made publicly (digitally) available.
- **Open Materials:** This badge is awarded when making available the materials of the followed research methodology necessary to reproduce or replicate that followed methodology (e.g. analysis scripts).
- **Preregistered:** That badge is awarded when preregistering a study design including the description of the research design and study materials.

[2] See https://www.acm.org/publications/policies/artifact-review-badging

How to award which badges depends on many (often non-trivial) criteria defined by editors and following a specific reviewing model to check the eligibility to obtain the badges. Although badges are, at the time of writing this chapter, rather rare in software engineering research (such as badges for preregistered studies) and although some systems may still be perceived as difficult to implement (such as the ACM system due to the wide spectrum of often overlapping badges), badges are generally recognised to be a valuable incentive that increases the participation in open science initiatives [26]. Hence, they are being adopted more and more by journals and conferences.

2.6 Open Peer Review

To the best of our knowledge, there is yet no commonly accepted and clear definition for open peer review. One definition focuses on the names of authors and reviewers and defines open review by both being known to each other. This allows for authors and reviewers to have a direct discussion rather than having to go through third parties for communication (e.g. via handling editors or chairs). In the programming community, this type of review process has long been known in code reviews, but – despite the advantages recognised in the research community as shown in a recent study on the future of peer review in software engineering [24] – it is not yet adopted by our journals and conferences (see also Sect. 5). One exception is the Journal of Open Source Software (for details see `https://joss.readthedocs.io/en/latest/submitting.html#the-review-process`). Another definition focuses on disclosing the reviews – sometimes with the names of the reviewers. That way, reviewers can be held more accountable, but they can can also serve to make the decision for acceptance more transparent to others and the reviewers can also claim the recognition they deserve.

3 Why do we need Open Science?

Open science is becoming more and more accepted in scientific communities having many positive effects. These effects range from increased access and citation counts [11] to facilitating technology transfer with the industry and fostering collaborations through open repositories. Academic publishing and knowledge sharing is meant to become more cost-effective – German university libraries alone are estimated to be spending well beyond 200 million EUR on publication subscriptions fees per year [28] – and researchers and practitioners with no publisher subscriptions can freely access and build on the work of others. There is, however, much more to it than the positive impact on institutions or individual authors, as explained next.

Imagine the following situation: A conference author submits a manuscript promising to have provided scientific and empirically-informed arguments for con-

sidering Go To statements harmful, a statement previously relying on rationalist arguments of software engineering pioneers like Dijkstra [10] only. As laid out by that author, those arguments emerge from the exploration of industrial source code – which the author does not share, maybe because of non-disclosure agreements with collaborating companies from which the data emerges, or maybe for other reasons, this statement is not made explicit in the manuscript – and from analysing the impact of those statements based on in-depth interviews – which the author does also not share, maybe because of ethical and legal constraints. Imagine further that the reviewers find no obvious methodological flaws in the design which the author describes in great detail for both the content analysis and the interviews. The author is an experienced and recognised authority in the research community and the manuscript is written in an easy-to-follow manner. The reviewers further find the manuscript "compelling", "interesting", and the results are also "surprising" to them given the availability of contrary evidence provided by other authors who previously analysed publicly available software repositories coming to very contrary conclusions [21]. Even if the submitting author did not discuss that other publication in detail, a presentation of that work would certainly lead to controversial and interesting discussions; something the reviewers believe to merit presentation at the prestigious conference they review for. So they recommend acceptance and the PC chairs select that publication for inclusion in the program. It is reasonable to believe that many readers of this chapter having served as co-chairs and reviewers for conferences can identify with such a situation.

Now imagine you were a young scholar analysing the effects of software defects and you find this publication. You would certainly find this publication interesting as it could provide a useful ground for follow-up work. Ask yourself – honestly – the following questions:

- Would you trust the results? If so, based on what? The simple fact that it has been accepted by the prestigious conference? The way the manuscript is generally written? The name of the author or her or his affiliation? Maybe based on the high number of citations that this publication already has? Maybe it is a combination of all factors? Would the picture change if the author would be unknown to you and if the work would have been published at a B-ranked conference?
- Would you be able to really comprehend how the study has been carried out? Would you be able to reproduce the conclusions drawn by the author based on the insights provided in the manuscript? Would you be able to replicate the study in your own research environment?
- To what extent does that piece of work provide a good theory for your work? Would this theory be robust and reliable (i.e. scientific)? Would you consider it useful?
- How would you use the work if you could only access the abstract of the manuscript because it is hidden behind a paywall and because your institution has no subscription? Would you cite based on the information in the abstract? Based on the statements found in other papers citing the work?

- How would you cite that work and put it in relation to your own research? Would it change in dependency to whether the statements in that manuscript support your own arguments or whether it contradicts them?

This very example certainly describes a fictitious situation and yet it describes in many ways the de-facto situation of software engineering research. Scientific practices are – and they need to – rely on certain safeguards such as peer review, but they are nevertheless also dictated by social and political mechanisms and many non-trivial, subjective factors in the research communities. These factors very often dictate in one form or the other which submissions eventually make it into the publication landscape and which do not, and which publications are cited and which are not. As a consequence, publication and citation regimes – although inherently rooted in scepticism – have also much to do with trust and convictions [8]; something which holds for most, if not all, scientific disciplines. Transparency is therefore key to break with scientific theories being grounded in common sense, taken-for-granted knowledge, hopes, convictions, and provisional beliefs.

Software engineering still faces many challenges other scientific disciplines do not face. Our data comprehends qualitative and quantitative data types and the theories we work on often have various disciplinary backgrounds (from mathematics over psychology to sociology). Further, our data very often emerges from highly sensitive environments making a disclosure difficult and in many cases impossible. Even if we can disclose the data, in many cases it has to be anonymised to an extent it becomes difficult to fully comprehend. All this renders building and evaluating theories in our field difficult. As an empirical, evidence-based discipline, we are thus still making our first steps and too often scientific practices are rooted in trust rather than being rooted in transparent scientific processes. Yet and as laid out in [8], it is theory building which constitutes a crucial foundation to our avenue towards turning our engineering discipline into a more scientific, evidence-based one as it was the case in many other disciplines before. Transparency, credibility, and reproducibility are cornerstones in building and evaluating robust and reliable theories for our still emerging field; and it is open science that provides the sole foundation to achieve that goal.

In essence, open science practices in general and data sharing in particular eventually allow us as a community of software engineering researchers and practitioners to effectively make contributions to our body of knowledge based upon shared data sets – making our empirical studies transparent, comprehensible, and credible – thus, we move forward as a community. As we argue, not only scientific publishing is essential in knowledge sharing and dissemination [17], but it is an essential facet in accumulating knowledge via a variation of studies tackling the same or similar questions and building upon the same or similar settings and data sets – e.g. as part of so-called replication studies [13] which are rendered difficult if not impossible without clear open science principles dictating shared values and principle scientific practices.

Therefore, there is no doubt anymore *whether* Open Science will become the norm also in software engineering research. Ever more public and private funding bodies are implementing open access and open data policies [6, 32]. Also the research

community is in tune with with this movement, as we can observe: editors and conference organisers are already planning for a smooth transition to open data, and reviewers are becoming more and more sceptical towards manuscript submissions which do not disclose their data and, consequently, ask the reviewers for too much credit. It remains, however, often still a question *how* the community could adopt open science practices and how individual researchers should open their research. We discuss this question in more detail in the next section.

4 How do we do Open Science?

In the following, we address the question of how to engage in open science. There are many aspects to consider when engaging as a researcher in open science. We believe that these aspects are best introduced along a simple (again, fictitious) scenario introduced next.

4.1 Exemplary Scenario

As an exemplary scenario, we consider a research project where we are researchers at European universities collaborating with project partners from other universities in the United States. Those partners are researchers in psychology. Our project aims at conducting a psychometric software engineering study and our overall goal is to collect data involving a large-scale study with human subjects. The research design is done in a joint effort. While our partners are largely responsible for the study execution and the data collection, we are largely responsible for analysing the data and reporting on it.

To keep the example simple, we focus on the statistical analysis of quantitative data, but also refer the reader to the challenges emerging from the disclosure of qualitative data in Sect. 5.

4.2 Overall Study Process

Figure 1 depicts, on the left side, the steps followed in our data analysis with a particular focus on those aspects relevant from an open science perspective. Overall, we first prepare our data and check for any errors, inconsistencies, and missing values, and we discuss these with our partners. At the same time, we start thinking about how to best answer our questions at hand. While we design our analysis procedure, we update the data structure to best fit the analysis plan. Once the analysis plan is finalised, we make it openly available. Ideally, we submit it as a *preregistered study*. This submission includes our study protocol and the material (analysis scripts) as

well as a detailed sample description allowing reviewers to judge upon the potential of the study with respect to its theoretical and practical impact. After registering our study and considering the feedback received, only then, we decide to begin with the data analysis.

After discovering no clear patterns in the data, we decide to participate at a conference where we present our ongoing work based on a previously published short paper describing the overall goal of the study and preliminary results. This work in progress presentation serves the purpose of receiving further feedback from the research community and of getting useful ideas on how to improve our data visualisation techniques. After successfully finishing our data analysis, we finally write up our main publication on the project and disclose our manuscript preprint prior to submitting our manuscript for review to a journal.

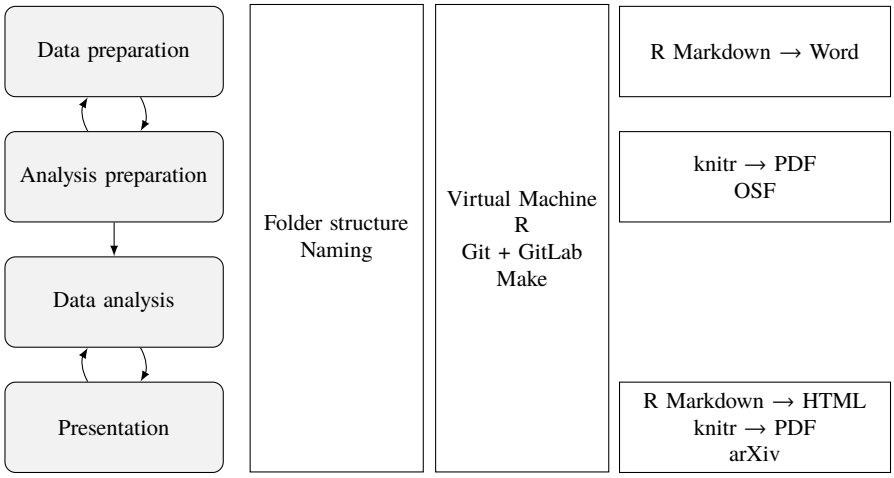

Fig. 1 Schema of an exemplary simple project.

In the following, we walk through that process while focusing on the infrastructure and tools. Our hope is that by presenting the process in such a pragmatic hands-on manner allows to fully reproduce the process as it should typically appear in a research setting.

4.3 Exemplary Walk-through

There are various tools to be used to make our project open and reproducible. While we do not claim to be able to present an exhaustive list here, our aim is to give some examples which we use ourselves to make recommendations based on our own experiences. One basic issue to consider first is the folder structure and the naming

convention. A good folder structure, in our view, could be like this as it captures the very essence of our process:

```
myproject/
├── README.md
├── Makefile
├── data/
│   ├── clean_data.Rmd
│   ├── clean_data.docx
│   ├── data_clean/
│   │   ├── mydata.csv
│   │   └── mymetadata.json
│   └── data_raw/
│       ├── messy_data1.xlsx
│       └── messy_data2.csv
├── analysis_plan
│   ├── analysis_plan.Rnw
│   └── analysis_plan.pdf
├── analysis/
│   ├── analysis.R
│   └── functions/
│       └── myfunction.R
├── conference_slides.Rmd
├── conference_slides.html
├── man_references.bib
├── manuscript.Rnw
└── manuscript.pdf
```

Note that the folder structure clearly defines the different steps shown in Figure 1 and the folder and file names clearly indicate what each of them contains.

Regardless of the actual size of the project, the basic rule should be to apply that structure and naming convention concisely and consistently. We experienced it to also be important to keep the original data in a separate folder (here data_raw/) and to not manipulate the raw data files but to create new data files in a separate folder for the data cleaning and analysis (data_clean/). In combination with a script which cleans the data (clean_data.Rmd), this makes the data cleaning process reproducible to others.

To keep the working environment stable in terms of software versions, we decide to use a *virtual machine* for this project. An alternative option could also be a container (Docker, Singularity, etc.). For the data cleaning and the analysis, we decide to use *R* [25], an open source software environment for statistical computing. An alternative to that could be to use Python. R scripts (e.g. analysis.R) are text files that can be executed in the R console. In contrast to click-and-point programs (e.g. SPSS when used without syntax) or programs producing binary files (e.g. Excel), R, same as Python, allows for a reproducible workflow which can be easily version controlled.

For version control, in our project, we decide to use *Git* [5] in combination with the Git-repository hosting service *GitLab* (`https://gitlab.com`). That version control system allows us and our collaborating partners to trace the versions of all produced text documents in an organised fashion. In combination with the hosting service GitLab, these versions remain available online to all involved in our project. For automating our workflow, we use Make [29]. To this end, we store a Makefile in our main project folder which contains the information on how different files depend on each other, for example that data/clean_data.Rmd depends on data/data_raw/messy_data1.xlsx and data/data_raw/messy_data2.csv and produces data/data_clean/mydata.csv, data/data_clean/mymetadata.json, and data/clean_data.docx. Our Makefile also documents how the outputs can be produced (via bash commands).

Next to using R for our project, we use *R Markdown* [37] and *knitr* [36]. Both allow users to combine R code chunks with explanatory text snippets and, thus, allowing for literate programming [19]. Our text is formatted with Markdown (R Markdown) and LaTeX (knitr). We regularly convert our R Markdown documents to Word documents to exchange them with our partners for constant feedback as they can comment on that document. This simplifies the communication about the constant data checking and cleaning process. For an intermediate project report and later for the manuscript writing, we use knitr as it gives us more formatting options.

Our analysis plan is written with knitr and we upload the PDF to the Open Science Framework (OSF, `https://osf.io`). This allows us to use the analysis plan for preregistration of the work we aim to do. Preregistration allows to reduce biases in the process of the data analysis (see also `https://osf.io/prereg`). We create the slides for the conference again using R Markdown which can produce high quality HTML slides. The manuscript is written using knitr and we make it available as open access on the preprint server arXiv (`https://arXiv.org`). To check whether preprint sharing is within the legal constraints of the publisher of the conference, we check for it using the search engine SHERPA RoMEO (`http://sherpa.mimas.ac.uk/romeo`).

As we see that the publisher follows a yellow open access model allowing to disclose the preprints but not the postprints, we choose to upload our preprint only. After that submission, we directly submit our manuscript to a peer-reviewed journal. Upon acceptance of the manuscript by that journal, we update our preprint with the DOI provided by the publisher, but do not submit the postprint, i.e, the post production version of the manuscript to comply with the copyright agreement. This preprint version is also the one we distribute among the community, e.g. via social media.

Since all root documents are text files (except for data/data_raw/messy_data1.xlsx) we can further put them under version control with Git. Through GitLab, we can make them easily accessible to others. This way, our project folder myproject/ can be seen as a replication package. Prior to disclosure, however, we check for parts in our data that need anonymisation to comply with the European data protection regulations as well as with the approval notification of the Institutional Review Board of

our partners in the U.S.. We remove any data that might allow to trace observations back to individuals participating in the study.

For our work to be reproducible in a long-term manner, we need to further document the versions of the software used. The virtual machine does that for us, but is not very portable. The option we follow is to use the version management system *packrat* in R [31].

We notice, however, that our partner is very reluctant to share the data because of its sensitivity and we thereby would not be able to follow the FAIR principle[3] as anticipated. It is, however, possible for us to convince our project partners to disclose the data when implementing some safeguards. To this end, we decide to disclose our data using the service platform Zenodo (`https://zenodo.org`) while chosing *Restricted Access*. Other researchers interested in accessing the data can first read the extensive meta data describing the content of the data and how it was produced. If they believe that the data would fit their scope of interest, they can apply for access and our previously established *Data Use and Access Committee (DUAC)*, formed by us data owners and a member of the responsible ethics committee, can decide whether we can grant access to the data or not.

That very example, we hope, illustrates an open science-conform study analysis and reporting producing all artefacts relevant to an open science format adoptable to software engineering and including the disclosure of:

1. A study protocol submission and review prior to publication (preregistered study)
2. The replication package including all analysed data (open data) and all files, scripts, and codebooks necessary to comprehend the study (open materials)
3. A preprint (yellow open access)

Needless to say, the example is a simplified one neglecting some challenges we typically encounter in practice. In the following, we discuss those challenges in more detail.

5 Challenges, Pitfalls and How-To's

In the following, we discuss typical challenges and pitfalls in open science from the perspective of researchers engaging in open science. To this end, we draw from our experiences covering both the roles of researchers and the roles of organisers (handling editors and conference and workshop organisers).

[3] FAIR stands for Findable, Accessible, Interoperable, and Reusable; see also `https://www.force11.org/group/fairgroup/fairprinciples`

5.1 General Issues

The major challenge that keeps researchers from following all the open science practices described above is probably the difficulty and effort required when making everything openly available. All the practices constitute additional steps that researchers have to do in addition to the non-open research process. They might be motivated to do these additional steps to support the scientific process and higher visibility of open publications. Yet, this motivation has limits. Therefore, the ease of doing open science practices is essential.

In our experience, the difficulty of being open has reduced dramatically over the years. It is easy and cost-free to handle a research project on GitHub or OSF, to permanently publish data on Zenodo or figshare, and to provide preprints on services like arXiv. Some difficulty lies in the details, such as the LaTeX requirements of arXiv, but nowadays we mostly work with modern web applications that behave as one would expect.

Another challenge that might keep researchers from employing openness in their research is the area of conflict between anonymity and confidentiality on the one side and openness on the other. In open science, we ideally would like to make everything open that helps others to understand, verify, and build on our work. When we work with companies, however, they have an understandable interest to protect their intellectual property and reputation, often reflected in signed non-disclosure agreements. Therefore, we have to reduce the data that we can make open or anonymise the data that we have. This is, again, additional effort and a risk that we accidentally make something open that should be confidential.

Similarly, when our studies involve humans, they have an interest in protecting their private data. With the EU GDPR, we now also have a strong legal basis for that. Hence, again, we have the risk to violate corresponding laws. In both cases, companies and individual humans, it is therefore imperative to publish any potentially sensitive data only with the explicit consent of the study participants. Only they themselves can decide what is sensitive and critical for them. In principle, this holds for any kind of publication and, hence, only needs to be extended to ask for consent for publishing the data as well. Anonymising company names is often enough. For anonymising sensitive data of study participants, there are also established techniques (see, e.g., [27]).

The challenge of anonymity also plays into the third more general issue we would like to mention: Often, openness is merely an afterthought. After we have done all the work, we provide a preprint and make the data available. Ideally, however, the whole process should be open, for example by using OSF or GitHub for all the documents, data, and analysis scripts. In terms of anonymity, this is difficult, as we cannot make everything open and often need a shadow repository with the original raw data. The raw data needs then to be carefully filtered when stored in an open repository. Yet, keeping everything open has the advantage that there is no way of manipulation during the analysis and publication phases of the research. We cannot make the hypothesis fit the data in hindsight because we documented the hypothesis before we did the analysis.

5.2 Sharing Preprints

For preprints, we need to consider where we want to publish the paper later on. Different publishers have different criteria about what they allow at all and what licences to chose.

Checking self-archiving options for Software Engineering

One helpful overview on the different self-archiving options in tune with the regulations of the major publishers in Software Engineering is, as we believe, provided by Arie van Deursen in [33].

Upon acceptance of our manuscript, we can post a postprint. This is rarely a problem when we already have a preprint that is simply updated. Otherwise, there might be publisher-specific embargo periods that need to be adhered to.

One challenge we would like to mention in context of preprint sharing emerges from a current trend in software engineering to push for double-blind reviewing models. While the goal is laudable to reduce potential biases by also anonymising the authors, it complicated open science practices considerably. Many conferences do not allow preprints to be made available because it might allow the reviewers to find out who the authors are. It has been our effort to start a trend in conferences to allow self-archiving of preprints and instruct peer reviewers to not actively look for the papers under review online, but it remains nevertheless a challenge.

The picture would change if open peer review would be implemented in a code review style (as discussed in Section 2.6). However, the downside and fear of many researchers is that open peer review will put a lot of pressure on researchers, especially early career researchers: Both as authors – the reviewers will know who made potential mistakes – and as reviewers – the authors will know who proposed the changes or even who recommended rejection of the paper.

5.3 Choosing Proper Licenses

A common pitfall while starting to use open science practices is to assign unsuitable licences. arXiv, for example, allows to select an ad-hoc non-exclusive license (to arXiv). Granting this minimal licence is compatible with any relevant venue a researcher might want to submit to. Hence, it keeps all options open even if the paper is rejected at the initially planned venue. Adding a Creative Commons licence could reduce this flexibility considerably. In fact, arXiv itself allows to chose from various Creative Commons licenses (CC BY, CC BY-SA, CC BY-NC-SA) as well as the CC0 dedication (i.e., public domain) [1].

Many argue that CC0 is preferable because it frees people from dealing with all attributions. However, in the scientific context, attributing the source and authors of all artefacts that are used is good practice independent of the licence used. PeerJ PrePrints, for instance, enforces the CC BY license exclusively [23]. This licence is also recommendable for postprints, provided postprint sharing is compatible with the publisher copyright agreement, as it ensures that the researchers are given credit while giving others the largest amount of freedom to share and reuse the manuscript.

In principle, choosing the proper licence is a non-trivial but important task, because certain licenses for preprints might cause incompatibility issues further down a publishing chain. Certain licenses, including some Creative Commons ones, prevent the work to be used in commercial settings (the -NC part of the CC) or require the redistribution of derivative works using the same license (the -SA part of the CC). Traditional publishers are, most of the times, commercial entities that require either a full copyright transfer or exclusive rights to distribute the work in a more restricted way, i.e. selling access to papers through paywalls. Non-commercial and share-alike CC licenses are, thus, in most of the cases incompatible with traditional publishing models.

Even the more liberal CC BY license, which only requires attribution and does not enforce a share-alike clause, might pose issues with traditional publishing as it is non-revocable and allows commercial use by anyone (i.e. non-exclusive to the publisher). The CC0 dedication has also caused issue with traditional publishing in the past [22]. The default licence by arXiv is a non-exclusive license to distribute [2], and, virtually, does solely allow arXiv to distribute and display a document (meaning that, theoretically, we are not allowed to do anything at all with arXiv submissions but reading them). This license is perhaps the most restrictive one among the free licenses, making it compatible with traditional publishing (if the copyright transfer conditions allow for it, see Section 2.1).

We can provide two recommendations. arXiv default non-exclusive license to distribute should be used when there is certainty to publish a paper with a traditional publisher. A CC-BY license should be used when there is certainty to publish a paper with a gold open access journal. We do not recommend licensing any preprint, postprint, or dataset using a non-commercial clause (-NC). While counter-intuitive at first sight (we wish for our work to stay free, after all), a non-commercial clause prevents the work to be used by commercial entities. The term *commercial* is, from a legal perspective much broader than it might appear at first; it might affect a large spectrum of people and entities including a simple blog if the website uses an advertisement system. There exist open companies that were born from commercial entities and that are therefore not non-profit (e.g., figshare and PeerJ), and these would not be allowed to make any use of material licensed with the -NC clause. Some of the work might include data mining of papers and datasets and aggregating results, which might still be very useful for the advancement of knowledge. For more information on these legal aspects, we direct the reader to a joint group of copyright experts and Wikimedia [34].

5.4 Sharing Data and Materials

A common pitfall in publishing open data and open materials, e.g. as part of replication packages, is to use a personal or institutional website for quickly and easily making them available. It gives one a unique ID in the form of a URL. Yet, a challenge is that we cannot ensure that the URL stays valid and that the content stays on the website. Therefore, repositories such as Zenodo or figshare, providing a DOI and ensuring permanent archival, are much preferable.

There are small differences between the repositories, but both are recommendable. figshare is commercial but free to use, and its usability seems more polished than at Zenodo. Furthermore, figshare participates in data preservation mechanisms while Zenodo does not. The permanency of Zenodo is ensured, because it is financed by the European Union and run by CERN.

Similarly as with preprint sharing in context of double-blind reviewing models, the availability of open data and material would also reveal the authors' identity and, hence, is rendered complicated. While there is no easy solution to the problem of sharing preprints when following a double-blind reviewing model, open data repositories allow researchers now to publish data anonymously for review, thus, being compliant to restrictions imposed by such reviewing model. The authors of the data can then be made public after the paper is accepted. A set of instructions on how to share and archive open data and keep it compatible with double-blind review can be found in [14].

5.5 Preparing Qualitative Data

Qualitative data is usually the most difficult to share in a replication package, because it is most personal and most difficult to anonymise within legal and ethical constraints. A number is more abstract than spoken (transcribed) words by individuals. Ideally, we anonymise also qualitative data and publish it with the explicit consent of the participants. It is important to be open about it upfront to understand whether the participants will agree. Especially for qualitative data, it might regularly not be the case that we get the consent. Then, it is even more important that everything else including the coding schema and coding rules is open so that reviewers and other researchers can at least check the trustworthiness of the analysis process and understand how the authors have drawn their conclusions.

6 Conclusion

Open science describes the movement to render all artefacts born out of scientific research activities accessible. Openness in our research processes is important to move forward in building reliable and robust theories, thus, turning our discipline

into a more scientific one. As outlined in this chapter, we still face, however, various challenges other disciplines do not face. Despite those challenges of adapting open science to the software engineering context, we can still see that our research community is making great progress in that direction. We have ourselves either accompanied or fully implemented efforts to help the community opening up their research artefacts covering, inter alia:

- Encouraging and supporting authors of the International Conference on Software Engineering 2017 in disclosing their preprints to the public yielding an increase in the participation ratio from 53% in 2013, where preprint sharing was documented for the first time, to a ratio of 71% in 2017.
- Implementing an open science policy at various events covering, for instance:
  - International Workshop on Cooperative and Human Aspects of Software Engineering (from 2016 on)
  - International Conference on Product-Focused Software Process Improvement 2017
  - International Symposium on Empirical Software Engineering and Measurement (from 2018 on)
  - ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2019
  - International Conference on Software Engineering 2021

In the course of this endeavour, we have noticed very well that introducing open science into a research community is a difficult and sensitive task, also because open science is still confronted with prejudice, but also because many authors, despite their willingness to conform with such policies, do not know how exactly to follow such an initiative; i.e. it is often difficult to see what we should do and what we can do (also considering ethical and legal constraints).

This is also the reason why, as organisers, we are often constraint by a general reluctance of implementing mandatory open science principles (e.g. via open data policies), thus, rendering the transition to more openness in our discipline rugged. However, the implementations of open science policies in recent editions of conferences – even if non-mandatory ones were authors could participate on a voluntary basis with the support of dedicated open science chairs – showed, nevertheless, a high participation ratio with more than 50% of the authors disclosing their data. Such participation ratios of authors and the positive feedback provided by the conference attendees, e.g. via Town Hall meetings, strengthens our confidence in that the research community is showing more and more awareness of the importance of open science and that open science will eventually become the norm.

One hope we associate with our ongoing efforts in implementing open science initiatives, such as the current open data initiative at the Empirical Software Engineering journal[4] – the first one to implement an open data initiative following a holistic process including a badge system – is also to send a strong signal into the software engineering research community to move further in that direction.

[4] https://github.com/emsejournal/openscience

Arguably, we are still confronted with various challenges, such as:

- How to implement a uniform and transparent guideline to review disclosed artefacts covering all possible variations in the different types of study (e.g. quantitative and qualitative ones)?
- How to implement preregistered studies (which we consider especially important to tackle the problem of publication bias) in tune with the reviewing processes of our existing journals and conferences and how to re-define existing roles and responsibilities?
- How to properly reward authors with a clear and easy to understand (and to use) badge system which recognises the differences in the various study types and the difficulties in opening up sensitive, e.g. industrial, data?
- How to implement open peer reviews. We can nowadays observe a significant turn in the existing single-blinded reviewing regime, which we applaud, but instead of opening up reviews as well, the current trend is towards even more closeness via double-blind reviewing models, thus, rendering other open science activities difficult, too.

Still, we are convinced that it is not anymore a question whether open science will become the norm also for the software engineering research community, but we recognise that there is still a long way to go, also because we still need to increase the awareness for what open science is, why it is so important, and how to properly adopt such principles to software engineering.

The book chapter at hands is intended to address these questions and to contribute to the movement. One hope we associate with this manuscript is to further encourage all members of our research community in joining us in this important endeavour.

Acknowledgements We want to thank all members of the empirical software engineering research community actively supporting the open science movement and its adoption to the software engineering community. Just to name a few, Robert Feldt and Tom Zimmermann, editors in chief of the Empirical Software Engineering Journal, are committed to push to implement a new Reproducibility & Open Science initiative for the major journal of the empirical software engineering research community. Markku Oivo, general chair of the International Symposium on Empirical Software Engineering and Measurement 2018, has actively supported the establishment of a new data sharing policy at the major Empirical Software Engineering conference so that we could pave the road for a long-term change. Natalia Juristo, general chair of the International Software Engineering Conference 2021, further actively supports the establishment of an open science initiative in our major general software engineering conference.

References

1. Arxiv. arxiv license information. `https://arxiv.org/help/license`, 2019. Archived: http://web.archive.org/web/20190410151011/https://arxiv.org/help/license. Accessed: 2019-04-10.
2. Arxiv. arxiv license information. `https://arXiv.org/licenses/nonexclusive-distrib/1.0/license.html`, 2019. Archived: http://web.archive.org/web/20190410165523/https://arxiv.org/licenses/nonexclusive-distrib/1.0/license.html. Accessed: 2019-04-10.

3. SÃűren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *DBpedia: A Nucleus for a Web of Open Data*, page 722âĂŞ735. Springer Berlin Heidelberg, 2007.
4. BOAI. Budapest open access initiative, 2002.
5. Scott Chacon and Ben Straub. *Pro git*. Apress, 2014.
6. Sue Childs, Julie McLeod, Elizabeth Lomas, and Glenda Cook. Opening research data: issues and opportunities. *Records Management Journal*, 24(2):142–162, 2014.
7. FOSTER Consortium. Open science taxonomy, 2019.
8. D. Méndez Fernández, J.-H. Passoth. Empirical Software Engineering: From Discipline to Interdiscipline. *Journal of Systems and Software*, 148:170–179, 2018.
9. Kay Dickersin. The existence of publication bias and risk factors for its occurrence. *JAMA: The Journal of the American Medical Association*, 263(10):1385, March 1990.
10. E.W. Dijkstra. Go To Statement Considered Harmful. *Communications of the ACM*, 2968.
11. Gunther Eysenbach. Citation advantage of open access articles. *PLoS biology*, 4(5):e157, 2006.
12. Paul Ginsparg. It was twenty years ago today... *arXiv preprint arXiv:1108.2700*, 2011.
13. Gómez, O.S. and Juristo, N. and Vegas, S. Replication Types in Experimental Disciplines. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2012.
14. Daniel Graziotin. How to disclose data for double-blind review and make it archived open data upon acceptance. `https://ineed.coffee/5205/`, 2019. Archived: https://web.archive.org/web/20190410141340/https://ineed.coffee/5205/. Accessed: 2019-04-10.
15. Daniel Graziotin, Xiaofeng Wang, and Pekka Abrahamsson. A framework for systematic analysis of open access journals and its application in software engineering and information systems. *Scientometrics*, 101(3):1627âĂŞ1656, 2014. Available: https://arxiv.org/abs/1308.2597.
16. Megan L. Head, Luke Holman, Rob Lanfear, Andrew T. Kahn, and Michael D. Jennions. The extent and consequences of p-hacking in science. *PLOS Biology*, 13(3):e1002106, March 2015.
17. John W Houghton and Charles Oppenheim. The economic implications of alternative publishing models. *Prometheus*, 28(1):41–54, 2010.
18. Norbert L. Kerr. HARKing: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2(3):196–217, August 1998.
19. Donald Ervin Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.
20. Craig Lambert. The Marketplace of Perceptions. *Harvard Magazine*, 108(4), 2006.
21. M. Nagappan, R. Robbes, Y. Kamei, E. Tanter, S. McIntosh, A. Mockus and A. Hassan. An Empirical Study of goto in C Code from GitHub Repositories. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015.
22. Russel O'Connor. The acm and me. `http://r6.ca/blog/20110930T012533Z.html`, 2011. Archived: http://web.archive.org/web/20190410153103/http://r6.ca/blog/20110930T012533Z.html. Accessed: 2019-04-10.
23. PeerJ. Peerj policies and procedures. `https://peerj.com/about/preprints/policies-and-procedures/`, 2019. Archived: http://web.archive.org/web/20190410151022/https://peerj.com/about/preprints/policies-and-procedures/. Accessed: 2019-04-10.
24. Lutz Prechelt, Daniel Graziotin, and Daniel Méndez Fernández. A communityâĂŹs perspective on the status and future of peer review in software engineering. *Information and Software Technology*, 95:75–85, 2018.
25. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.
26. Anisa Rowhani-Farid, Michelle Allen, and Adrian G Barnett. What incentives increase data sharing in health and medical research? a systematic review. *Research integrity and peer review*, 2(1):4, 2017.
27. Benjamin Saunders, Jenny Kitzinger, and Celia Kitzinger. Anonymising interview data: challenges and compromise in practice. *Qualitative Research*, 15(5):616–632, 2015. PMID: 26457066.

28. Ralf Schimmer, Kai Karin Geschuhn, and Andreas Vogler. Disrupting the subscription journalsâĂŹ business model for the necessary large-scale transformation to open access. 2015.
29. Richard M Stallman, Roland McGrath, and Paul Smith. *GNU make*. Citeseer, 2001.
30. Jonathan Tennant, Jennifer E Beamer, Jeroen Bosman, BjÃűrn Brembs, Neo Christopher Chung, Gail Clement, Tom Crick, Jonathan Dugan, Alastair Dunning, David Eccles, and et al. Foundations for open scholarship strategy development, Jan 2019.
31. Kevin Ushey, Jonathan McPherson, Joe Cheng, Aron Atkins, and JJ Allaire. *packrat: A Dependency Management System for Projects and their R Package Dependencies*, 2018. R package version 0.5.0.
32. Veerle Van den Eynden, Louise Corti, Matthew Woollard, Libby Bishop, and Laurence Horton. Managing and sharing data; a best practice guide for researchers. 2011.
33. Arie van Deursen. Green open access faq. `https://avandeursen.com/2016/11/06/green-open-access-faq/`, 2016. Archived: https://web.archive.org/web/20190410141222/https://avandeursen.com/2016/11/06/green-open-access-faq/. Accessed: 2019-04-10.
34. Wikimedia. *Consequences, risks and side-effects of the license module "non-commercial use only"*. OpenGLAM. 2013.
35. Michael Woelfle, Piero Olliaro, and Matthew H Todd. Open science is a research accelerator. *Nature Chemistry*, 3:745 EP–, Sep 2011.
36. Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. ISBN 978-1498716963.
37. Yihui Xie, J.J. Allaire, and Garrett Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. ISBN 9781138359338.