ENCORE. A practical implementation to improve reproducibility and transparency of computational research

Antoine H.C. van Kampen[1,2*#], Utkarsh Mahamune[1#], Aldo Jongejan[1], Barbera D.C. van Schaik[1], Dasha Balashova[1], Danial Lashgari[1], Mia Pras-Raves[1,3], Eric Wever[1,3], Rodrigo García-Valiente[1], Adrie D. Dane[1,3], Perry D. Moerland[1]

[1]Amsterdam UMC location University of Amsterdam, Epidemiology and Data Science, Meibergdreef 9, Amsterdam, Netherlands. Amsterdam Public Health, Methodology, Amsterdam, The Netherlands. Amsterdam Infection and Immunity, Inflammatory Diseases, Amsterdam, The Netherlands.

[2]Biosystems Data Analysis, Swammerdam Institute for Life Sciences, University of Amsterdam, Amsterdam, the Netherlands,

[3]Amsterdam UMC location University of Amsterdam, Lab. for Genetic Metabolic Diseases, Meibergdreef 9, Amsterdam, The Netherlands

# Equally contributed

*Corresponding author:
Antoine H.C. van Kampen
Bioinformatics Laboratory
Epidemiology and Data Science
Amsterdam University Medical Centers
Meibergdreef 9, 1105 AZ Amsterdam, the Netherlands
a.h.vankampen@amsterdamumc.nl
tel. +31-20-5667096

Abstract

Reproducibility of research outcomes has become increasingly important within all research fields, including the (bio)medical domain. Despite ongoing (open-science) initiatives, computational research is often still difficult to reproduce. Although many papers are available that layout clear guidelines to improve computational reproducibility, the translation and application of these guidelines into practice seems to be an obstacle. We propose ENCORE (ENhancing COmputational REproducilbity) as a practical implementation of previously published guidelines to improve reproducibility and transparency through a tight integration of all project components (data, code, results, concepts, documentation). ENCORE comprises a standardized file system structure (sFSS), pre-defined (Markdown) files for documentation, a GitHub repository for software version control, a HTML-based sFSS browser (FSS Navigator), and ENCORE documentation, to set up a self-contained project package and support researchers to use ENCORE in a consistent manner. ENCORE can be used for any type of computational project. It improved reproducibility and transparency of our computational projects but, at the same time, made clear that achieving full reproducibility requires additional steps. Perhaps the most significant challenge towards improved reproducibility is the lack of clear and direct incentives for and intrinsic motivation of the scientists.

Keywords: Computational research, Reproducibility, Transparency, Open-Science, Open-Source, Open-Access, GitHub.

Introduction

Reproducibility of research outcomes has become increasingly important within all research fields, including the (bio)medical domain, which is also the focus of this paper (Ioannidis, 2005; Stupple et al., 2019) . The scientific community is increasingly aware of a reproducibility crisis with many studies that still have to be reproduced by independent researchers or that could not be reproduced when attempted (Fidler & Wilcox, 2021). Reproducibility is an important hallmark for scientific research since public trust, new knowledge and projects should be based on established and proven principles (Diaba-Nuhoho & Amponsah-Offeh, 2021). Irreproducible research is nothing more than a miracle, and has no place in science (Markowetz, 2015). Larger datasets, increased complexity of experimental and computational methods, and multi-disciplinarity, make reproducibility increasingly challenging. Scientist-driven efforts complemented with pressure from research institutes, funders, and journals resulted in various initiatives and approaches to increase reproducibility of scientific findings, covering the different stages of the research lifecycle (Turkyilmaz-van der Velden et al., 2020) (**Figure 1**). Typically, this cycle starts (stage 1) with a careful preparation of the study, which includes planning in terms of research objectives, funding, data management, ethics, experimental and computational approaches, publishing, and study archiving. The next stage concerns the collection and processing of (patient) samples, and the generation of data. Once data has been collected, it is analyzed using statistical or other computational methods. Mathematical modelling and simulations can be part of this computational phase. Occasionally, research make use of existing data that is available from public repositories (CERN & OpenAIRE, 2023; Rigden & Fernandez, 2023), or are completely based on computer simulations (Lashgari et al., 2022; Merino Tejero et al., 2020). This may complement the data collection during stage 2. In the final stage, research outcomes are published in open-access preprint repositories (e.g., bioRxiv, medRxiv), and peer-reviewed (open-access) journals, and archived in repositories such as figshare (Figshare, 2023) and Zenodo (CERN & OpenAIRE, 2023). Publication and archiving should also include any data that was generated and software that was developed. Open Science initiatives aim for open-access publications and data (Pulverer, 2018; van der Heyden & van Veen, 2018) as well as open-source software (Garijo et al., 2022; Open Source Initiative, 2023).

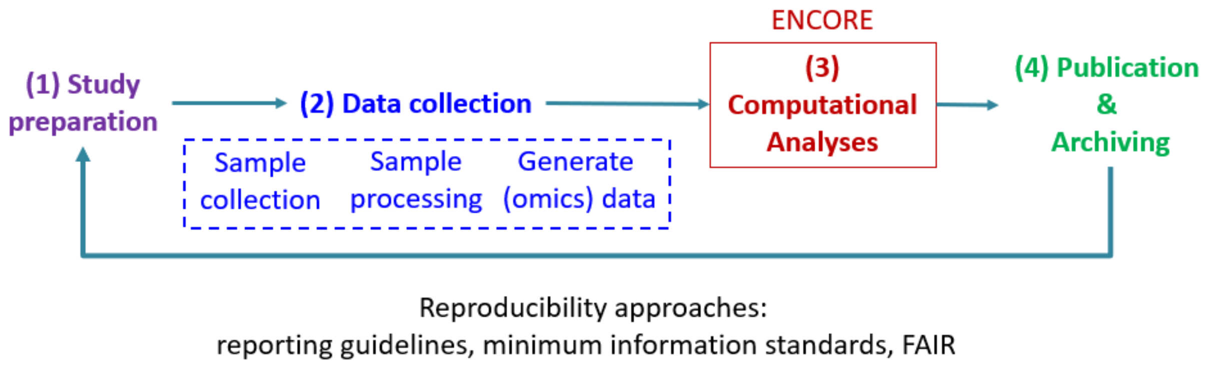

ENCORE
(1) Study preparation
(2) Data collection
Sample collection
Sample processing
Generate (omics) data
(3) Computational Analyses
(4) Publication & Archiving
Reproducibility approaches:
reporting guidelines, minimum information standards, FAIR

Figure 1. Research lifecycle. Most (clinical and biomedical) studies go through the stages of (1) study preparation, (2) data collection (involving sample collection and processing, and data generation), (3) computational analysis, and (4) publication & archiving. Subsequently, based on obtained results, follow-up studies are designed. At every stage and between stages, multiple iterations may occur. A range of approaches have been suggested to improve study reproducibility for the different stages. ENCORE focuses on computational reproducibility (Stage 3).

Computational analyses (Stage 3; **Figure 1**) can be carried out using existing (commercial) software applications, but it increasingly involves the development of new software by the scientific community. It is crucial that newly developed software is tested, well-documented, and becomes available as open-source to allow peers to use, inspect, verify, modify, and enhance the software (Hunter-Zinck et al., 2021; Karimzadeh & Hoffman, 2018; Margan & Čandrlić, 2015). The increasing use of systems like Git and GitHub in (bio)medical scientific communities significantly improved software version control, hosting, and sharing, thereby facilitating transparency and reproducibility of computational research (Blischak et al., 2016; Deardorff, 2020; Perez-Riverol et al., 2016; Ram, 2013).

Despite ongoing initiatives to improve computational reproducibility, results from computational analyses are often difficult to reproduce even if data and code are available (Mendes, 2018; Papin et al., 2020; Stodden et al., 2018; Tiwari et al., 2021). This also holds for our own computational analyses. Irreproducibility of computational research has well-known causes, including unavailability of software, changes in algorithms with updates of software libraries, incomplete software documentation, and missing parameters, to mention a few. Many papers are available that layout clear guidelines to improve computational reproducibility (e.g., (Sandve et al., 2013; Wilson et al., 2017; Ziemann et al., 2023)). However, translation and/or application of these guidelines into research practice seems to be a bottleneck given that still a large portion of computational studies do not adhere to these guidelines and are difficult to reproduce. Moreover, in our experience, one main cause is that software and data are decoupled and archived in different repositories. Nowadays, there is a large focus on (FAIR; (Wilkinson et al., 2016)) data management to ensure data availability, and the use of software repositories such as GitHub to make software available. However, this adds another layer of complexity, leaving it up to the researcher to determine how (subsets of) the data were used with the software, and how this connects with the results. This may not always be obvious from the publication. This in general leaves it to the researcher to reconstruct the precise computational workflow to reproduce results. In addition, lack of (software) documentation and undocumented manual steps, further complicate matters. Consequently, computational reproducibility can only improve if data, code, documentation, and (intermediate) results are much more tightly, consistently, and transparently integrated, and if project documentation itself significantly improves. Such a self-contained project package could also easily be shared with peers and reviewers.

In this paper we focus on computational reproducibility, that is the reanalysis of the same data using the same computational methods. We propose ENCORE (ENhancing COmputational REproducilbity) as a practical implementation of previously published guidelines, resulting in a tight integration of all project components. ENCORE guides researchers to structure and document a project according to established guidelines to improve transparency and reproducibility. ENCORE therefore harmonizes the approach to improve computational reproducibility within a research group. The development of ENCORE started in 2018 and is based on existing ideas and initiatives (Brito et al., 2020; Sandve et al., 2013; Spreckelsen et al., 2020; Turkyilmaz-van der Velden et al., 2020), discussions within our research group, and improvements made over a five-year period. ENCORE does not specifically address stages 1, 2 and 4 of the research lifecycle (**Figure 1**). However, (parts of) the documentation resulting from the other stages including a description of the study preparation (e.g., hypotheses to be tested, experimental design), the (patient) samples, the physical experiment, and the (FAIR) data should be included in the project package for completeness but also to correctly design and perform the computational analyses (Stodden, 2015). Conversely, the documentation of computational protocols facilitates the documentation of (pre)processed data to meet the FAIR requirements. In addition, it provides (supplementary) information for research papers. Here, we describe the design of ENCORE, our internal evaluations of its usage, and our experience in using it for research and for support projects that we carry out for other research groups. ENCORE improved reproducibility and transparency of our computational projects but, at the same time, made clear that achieving full reproducibility requires additional steps. Perhaps the most significant challenge towards improved reproducibility is the lack of clear and direct incentives for and intrinsic motivation of the scientists.

Methods

ENCORE comprises a standardized file system structure (sFSS), pre-defined (Markdown) files for documentation, a GitHub repository for software version control, a HTML-based sFSS browser (FSS Navigator), and ENCORE documentation, to set up a self-contained project package and support the researcher to use ENCORE in a consistent manner. The sFSS template from the ENCORE GitHub repository (https://github.com/EDS-Bioinformatics-Laboratory/ENCORE). ENCORE can not only be used for research projects but also for support projects in which computational analyses are performed for third parties on request, or for education (e.g., computer tutorials).

The implementation of ENCORE was driven by a set of basic requirements:

ENCORE approach was driven by several main requirements:

1. **Single self-contained project package**. The computational project should be organized and available as a self-contained and integrated package of data, code, results, concepts, and documentation, stored at a single location, and which is easily transferable to other researchers or reviewers without breaking its internal consistency such that the code remains executable.
2. **Facilitate transparency and documentation**. ENCORE should facilitate transparency and a deep understanding (e.g., addressing why specific methods were selected and how these were applied) of the project through its standardized structure and documentation of concepts, methodology, data, code, and results.
3. **Enable reproducibility**. The project package should enable an external researcher to autonomously execute and understand the computational techniques and recreate the (published) outcomes.
4. **Adhere to established guidelines**. ENCORE should adhere to published guidelines for computational reproducibility.
5. **Enable version control**. ENCORE should allow version control of code and code documentation.
6. **Facilitate harmonization**. The ENCORE approach itself should be standardized and well-documented such that it can easily be adopted by any researcher. This allows harmonization within research groups, enabling the further joint development of best practices within the ENCORE framework. Moreover, harmonization also facilitates checking transparency and reproducibility prior to publication by direct colleagues.
7. **Provide a generic approach.** ENCORE should be agnostic for the type of computational project (e.g., statistical analysis, mathematical modelling), data, programming language, and ICT infrastructure (e.g., operating system and computer hardware). It should not depend on (project management) tools that are not freely available.

8. **Allow adaptation to style of work**. ENCORE should leave sufficient flexibility to accommodate different styles of work. It should be accessible from any software tool the researcher might be using.

The sFSS in context

The sFSS is the project's entry point (**Figure 2**) and, therefore, should be self-contained, which is the responsibility of the project team. The software developed in the project and the external documentation are also made available through Git and the project's GitHub repository to enable version control and joint development. The principled decision to only make code and code documentation available on GitHub and exclude other project components like data and results, is motivated by the requirement to create a self-contained sFSS project package and the fact that GitHub is not a universal storage platform. Consequently, in practice, an sFSS contains only the software version that will be shared, while the GitHub repository may also contain older or alternative versions not required to reproduce the computational results. In addition, also the size of the data and results may be too large to host on GitHub. The complete project package can be shared with other researchers or reviewers, or submitted to public repositories such as Zenodo for archiving, in which case a DOI can be assigned. Providing access to the GitHub repository is optional. Sharing of support projects that are carried out for non-computational researchers (e.g., clinicians) who are mostly or only interested in the results and who will not attempt to reproduce the results may require reducing and/or restructuring the sFSS. For example, one may flatten the directory structure, remove the code, and/or reorganize the presentation of the main results.

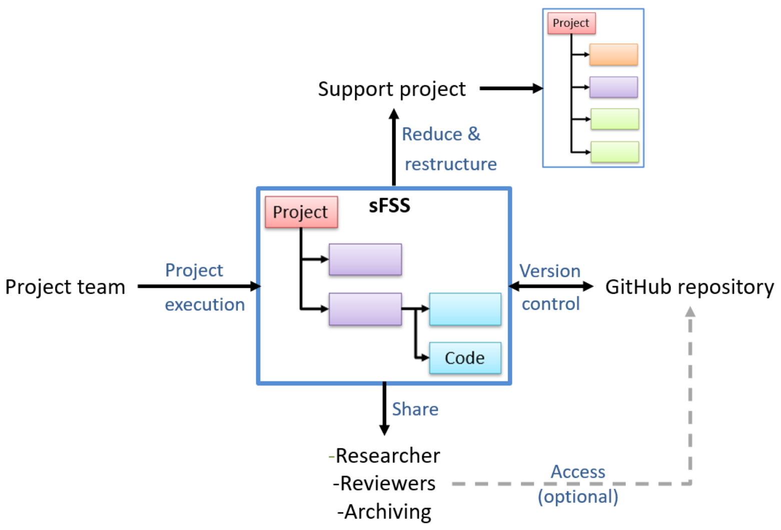


Figure 2. The sFSS and its environment. The sFSS is the central point of entry for a project. The project team is responsible for the organization and documentation of the project. Only the code and code documentation are synchronized to the project's GitHub repository. An sFSS project package can be shared or archived. For support projects, the sFSS can be reduced and/or restructured prior to sharing with non-computational researchers.

Instantiation of a new ENCORE-based project

ENCORE enables several approaches to set up a new project. One first creates a new project GitHub repository. Subsequently, one clones the sFSS template from the ENCORE GitHub repository using Git bash and connect it to the project’s GitHub repository. Finally, one starts documenting the project. This approach is documented in the ENCORE Step-by-Step guide (**Supplementary File 1**) and takes approximately 30 minutes to complete. This will be the approach for most research projects. Another option is to generate a small program (script) that retrieves the sFSS template from the ENCORE GitHub repository and automatically fills in certain portions of the required documentation, which is particularly useful for routine support projects. For specific cases, one may also write software that does not use the sFSS template available on GitHub but generates the structure and files itself. However, this is not recommended since such software can easily get outdated with further modifications to the sFSS template.

The ENCORE components

ENCORE comprises five main components (**Figure 3**) to structure, integrate, control code and documentation versions, and to improve transparency and reproducibility of a project.

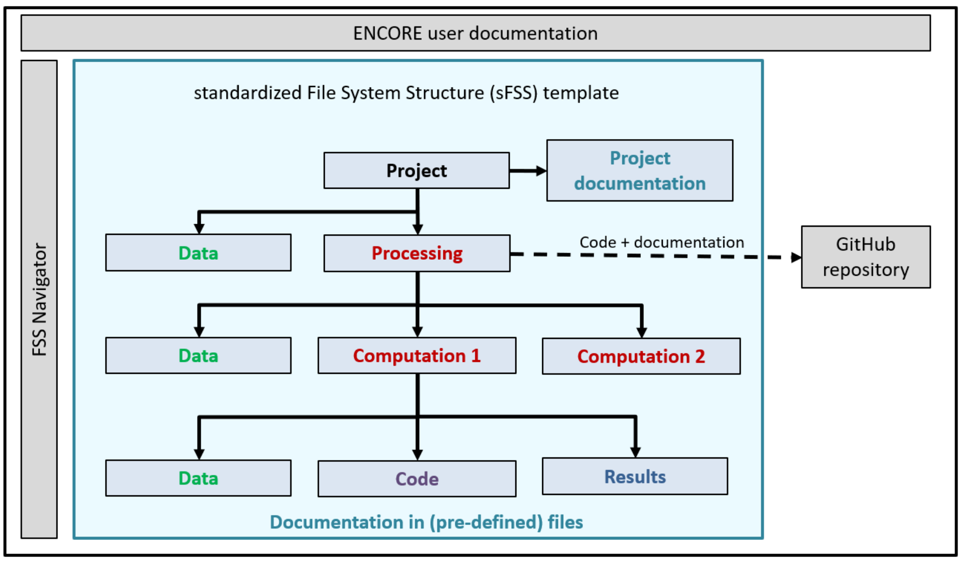


Figure 3. Components of ENCORE. The main component comprises the sFSS template (blue block) that organizes all parts of the project. ‘Project’ corresponds to the root directory of the template. The other blocks represent project dependent sub-directories. Each project is complemented with a GitHub repository for version control of the code and documentation in the ‘Processing’ (sub) directories. Project documentation resides in (pre-defined) files that are found in all subdirectories of the sFSS template. The pre-defined files contain instructions on the minimum information that needs to be provided for the documentation of the different parts of a project. In addition, the external ENCORE user documentation provides instructions for new users to instantiate a new project. The FSS Navigator allows (end) users to browse the main and specified parts of the project.

Component 1. The standardized File System Structure (sFSS) template

The sFSS provides a standardized, yet flexible, template to organize conceptual information, data, code, documentation, and (intermediate) results (e.g., tables, figures, text files). In essence, it is a standardized directory structure containing pre-defined files that can be used with any operating system (**Figure 4**) and can be used with any data analyses (or other) software that reads and writes to a file system. Conceptual information includes any information considered relevant to correctly setup the computations, but also information that helps peers to understand the (background of the) project and information that documents why specific choices were made. This includes information about the research question, experimental design, samples used, wet-lab experiments, description of computational (for example, statistical or mathematical modelling) approaches, interpretation of results, relevant literature, presentations about the project, and summaries from (email) discussions during the project. Items in a project may exist as different versions if, for example, new data was generated, code was modified, or figures were updated based on different settings for the computations. The information within the sFSS is implicitly and explicitly integrated. Implicit links correspond to relations that are imposed due to the hierarchical structure of the sFSS. Explicit links are made in the (code) documentation, for example, by linking a particular computation to a specific subset of the data. The sFSS allows a certain degree of flexibility to accommodate different types of projects or different ways of working. For example, data can be organized at different levels in the sFSS (**Figure 3**). This allows, for example, to organize the data directly within the subdirectory of a specific computation, if that data is not used by other computations. Similarly, if a project involves the (pre)processing of data, then the outcome of these steps may either be considered as a 'Result' in a 'Computation' directory, or as 'Processed data' in a 'Data' directory. The 'LabJournal' can be copied to other directories to accommodate more specific documentation. Markdown README files for project documentation may be replaced with other file types (e.g. LaTeX) if one prefers. Unused directories and files should be removed to reduce the complexity of the sFSS. Detailed information about specific directories and pre-defined files is found in **Supplementary File 2**.

```
ID_ProjectName
•  00_README-FIRST.{md, txt}
•  0_GETTINGSTARTED.{docx, tex, txt, html}
•  0_PROJECT.md
•  1_Step-by-Step-ENCORE-'   Guide.docx
•  2_CITATION.{md, txt}
•  Navigate.py / Test_Navigate_Module.py
•  Navigator executables (Windows, MacOS, Unix)
o  Data                    (0_README.md)
      •  NameOfDataset_1
            •  Meta        (0_README.md)
            •  Processed   (0_README.md)
            •  Raw         (0_README.md)
o  Processing              (README.md, github.txt, gitignore-templates)
      •  .git
      •  0_SoftwareEnvironment  (0_README.md)
            •  Anaconda          (0_README-General.md, 0_README-ProjectSpecific.md)
            •  C++
            •  Matlab
            •  Python
            •  R
      •  Data
            •   NameOfDataset_1
                  •  Meta
                  •  Processed
                  •  Raw
      •   NameOfComputation
            •   Code                    (0_README.md)
            •   CodeDocumentation       (0_README.md)
            •   Data
                  •  NameOfDataset_1
                        •  Meta
                        •  Processed
                        •  Raw
            •   NoteBooks              (0_README.md)
            •   Results                (0_README.md)
            •   Settings               (0_README.md)
o  ProjectDocumentation                (LabJournal.{docx, tex, md, txt})
      •  BackgroundDocumentation
      •  Literature                  (0_README.md)
      •  MyPresentations             (0_README.md)
o  Manuscript                        (0_README.md)
o  Sharing                           (0_README.md)
```

Figure 4. The standardized File System Structure (FSS) and associated pre-defined files. Standardized directory structure of the sFSS containing pre-defined files (brown), which include README files (in Markdown format) that provide a documentation template and instructions. Note that the pre-defined files in the 'Data' directories (green) and the

'0_SoftwareEnvironment' subdirectories are only shown once. The names of the directories 'NameOfDataset_1' and 'NameOfComputation' are placeholders and should be replaced with more descriptive names. These directories can be replicated if multiple datasets are used or if different computation procedures are performed. Subdirectories shown in blue are under version control using Git/GitHub. The '0' prefix ensures that the corresponding files/directories are always on top of the file list when using lexicographic ordering. The README.md in 'Processing' is the default GitHub repository README file and therefore does not have the '0' prefix.

Component 2. Pre-defined files

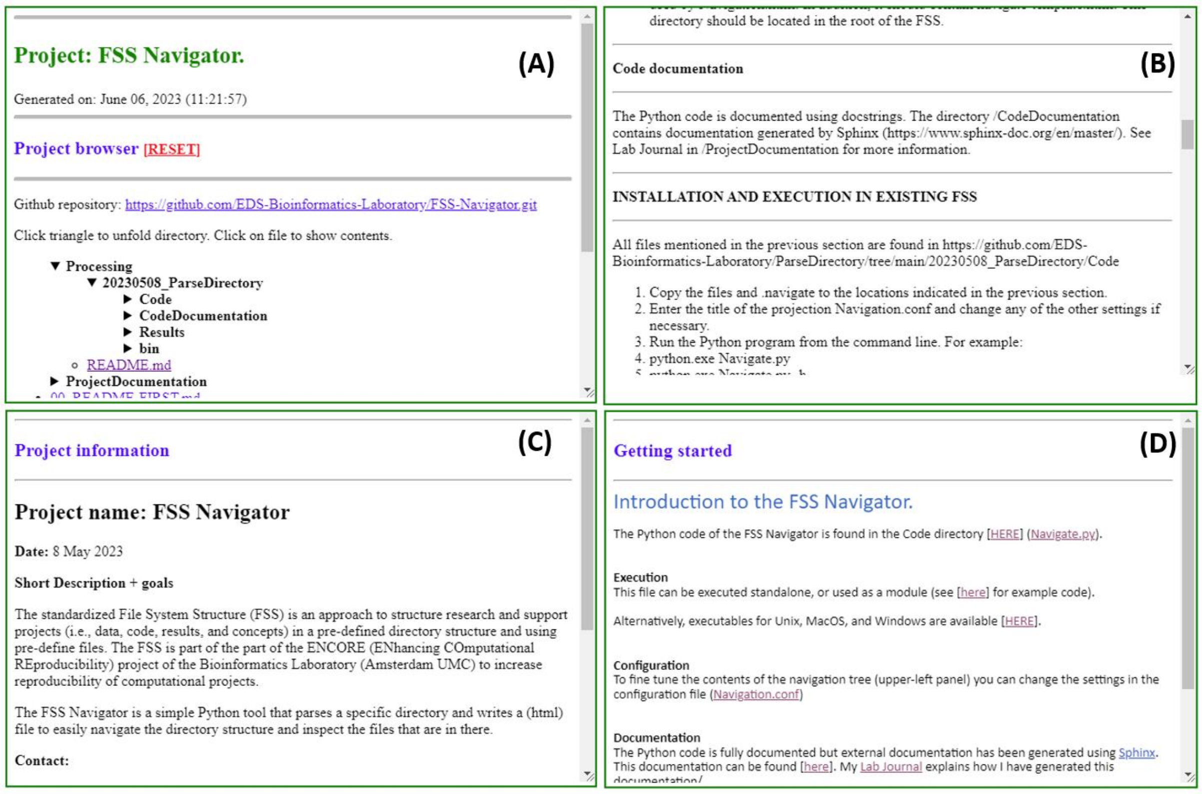
The sFSS contains many pre-defined files that should be used directly from the start of the project and should be maintained throughout the project. Most files are Markdown files. Markdown is a markup language, which enables adding formatting elements to plain text (Markdown, 2023) and requires a compatible editor. However, if preferred, these may be replaced by other file formats (e.g., LaTeX, Microsoft Word). In the sFSS root directory, the *0_PROJECT.md* file should provide basic project details, including the project name, start date, a short description, and project team. In addition, the *0_GETTINGSTARTED.html* file should describe the most important aspects of the project and should provide links to the relevant sub-directories and files. 'Getting started' templates are provided in different file formats, which can be converted to HTML once finished. The *0_PROJECT.md* and *0_GETTINGSTARTED.html* files are used by the FSS Navigator (see below). The *LabJournal* file in *ProjectDocumentation* should contain general project documentation including but not limited to an explanation of the project's background and concepts, computational approaches, summaries of project discussions, new research ideas, and to-do lists. Preferably, the lab journal should contain pointers to relevant sub-directories and files whenever needed. Alternatively, one may maintain multiple lab journals in different directories containing documentation for, *e.g.*, a specific computation or dataset. The lab journal is also important for the scientific legacy of the research group by ensuring that others can replicate what the original researcher(s) has done. We decided to deviate from standard practice and also use the lab journal to record new ideas, provide summaries of (email) discussions, and to-do lists, since it is important to have a record of these for the supervision of the projects and for follow-up projects. Consequently, not all information in the lab journal can be shared with others (see Discussion). In addition to the lab journal, each directory contains a Markdown README file, which is specific for the directory in which it is located. In general, these README files contain an explanation of the information found in a specific directory, instructions, and a documentation template specifying the minimum required documentation that a researcher should provide. The instructions and templates are basically a translation of published guidelines to enhance reproducibility of computational projects. This is an essential part of ENCORE, since following the instructions and templates will significantly improve the reproducibility of the project and prevents the researcher from re-inventing the wheel or reading the many publications about computational reproducibility. The instructions and templates also enforce internal consistency of an sFSS and promotes consistency between sFSS within a research group. An example of a README file is given in **Supplementary File 3**. One decision to be made by the project team is how to distribute the documentation over the various README files and the lab journal(s). However, as a rule one should document any project item (code, data, results) in the directories in which these are located. The lab journal can then be used for more general documentation.

Component 3. The GitHub repository

ENCORE utilizes Git and GitHub for version control of software and its documentation, and to collaborate on software development. Since the sFSS project package contains all information of a project, it is not necessary to also share the GitHub repository. However, the project team may still share this repository or make it public in case of joint code development or if access to previous code versions is requested or required. To ensure that only code, (Jupyter) notebooks, software settings, and documentation is managed by Git, the project owner needs to configure the so-called '.ignore' file. A template is provided in the sFSS, which can be adapted and complemented with pre-existing templates depending on the programming language and environment used (**Supplementary File 1**) (GitHub, 2023).

Component 4. The FSS Navigator

At the end of a project, an sFSS may contain a large amount of information, potentially making it difficult for peers to determine the optimal point of entry. Thus, FSS Navigator was developed to provide a first overview of the project. It also provides a convenient tool during the project to, for example, browse and show results. The FSS Navigator itself was developed following the ENCORE approach, and the project package can be found in Zenodo (Van Kampen et al., 2023). The Navigator is a Python program, which scans the sFSS and generates a web page (Navigate.html) that can be opened in any web browser. In addition to the Python program, executables for Windows, macOS (for both Intel- and silicon-based chips) and Linux/Unix are provided to ensure the Navigator can be used if Python is not installed. The generated web page consists of four panels (**Figure 5**). The content of the panels is configured by the project owner, which allows to guide peers to the important parts of the project.



(A)

Project: FSS Navigator.

Generated on: June 06, 2023 (11:21:57)

Project browser [RESET]

Github repository: https://github.com/EDS-Bioinformatics-Laboratory/FSS-Navigator.git

Click triangle to unfold directory. Click on file to show contents.

```
▼ Processing
    ▼ 20230508_ParseDirectory
        ▶ Code
        ▶ CodeDocumentation
        ▶ Results
        ▶ bin
  ○ README.md
▶ ProjectDocumentation
```

(B)

directory should be located in the root of the FSS.

Code documentation

The Python code is documented using docstrings. The directory /CodeDocumentation contains documentation generated by Sphinx (https://www.sphinx-doc.org/en/master/). See Lab Journal in /ProjectDocumentation for more information.

INSTALLATION AND EXECUTION IN EXISTING FSS

All files mentioned in the previous section are found in https://github.com/EDS-Bioinformatics-Laboratory/ParseDirectory/tree/main/20230508_ParseDirectory/Code

1. Copy the files and .navigate to the locations indicated in the previous section.
2. Enter the title of the projection Navigation.conf and change any of the other settings if necessary.
3. Run the Python program from the command line. For example:
4. python.exe Navigate.py

(C)

Project information

Project name: FSS Navigator

Date: 8 May 2023

Short Description + goals

The standardized File System Structure (FSS) is an approach to structure research and support projects (i.e., data, code, results, and concepts) in a pre-defined directory structure and using pre-define files. The FSS is part of the part of the ENCORE (ENhancing COmputational REproducibility) project of the Bioinformatics Laboratory (Amsterdam UMC) to increase reproducibility of computational projects.

The FSS Navigator is a simple Python tool that parses a specific directory and writes a (html) file to easily navigate the directory structure and inspect the files that are in there.

Contact:

(D)

Getting started

Introduction to the FSS Navigator.

The Python code of the FSS Navigator is found in the Code directory [HERE] (Navigate.py).

Execution
This file can be executed standalone, or used as a module (see [here] for example code).

Alternatively, executables for Unix, MacOS, and Windows are available [HERE].

Configuration
To fine tune the contents of the navigation tree (upper-left panel) you can change the settings in the configuration file (Navigation.conf)

Documentation
The Python code is fully documented but external documentation has been generated using Sphinx. This documentation can be found [here]. My Lab Journal explains how I have generated this

Figure 5. The FSS Navigator. (A) Expandable sFSS directory tree and link to the project's GitHub repository. The project owner can configure, which directories and files to show. (B) Content of a selected file. In this example, the panel shows the content of the default GitHub Markdown README file. (C) General project description, contact person, and collaborators (0_PROJECT.md). (D). Getting started explains the project and directly includes links to the various files and directories in the sFSS (0_GettingStarted.html).

Component 5. User documentation

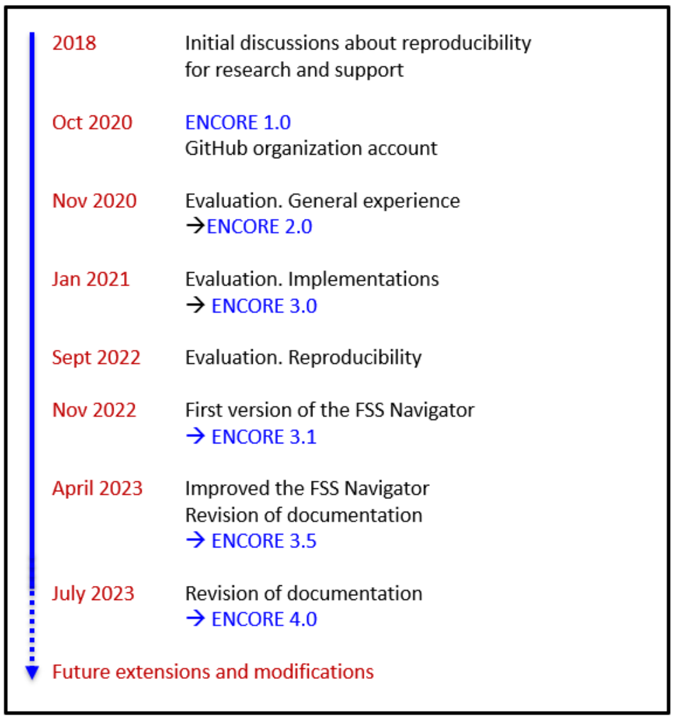
ENCORE is complemented with extensive documentation to guide the user in setting up a new project and maintaining documentation throughout the project. In addition to the documentation already present in the pre-defined files, we have created a comprehensive 'Step-by-Step ENCORE Guide' (Supplementary File 1). This guide offers a brief introduction to the ENCORE principles and components and provides a recipe for instantiating an sFSS for a specific project, a corresponding GitHub repository, and FSS Navigator. It also includes a basic introduction to Git and GitHub for troubleshooting purposes.

General ENCORE guidelines

The initialization of an ENCORE-based project is straightforward and does not consume much time. However, it is important to keep working according to the provided instructions and to keep the documentation continuously up to date. Recollecting (from memory) all important details at the end of a project is virtually impossible. Moreover, this will take significantly more time than gradually adding documentation during the project. While organizing and documenting a project, one should assume that at some stage the project is shared with a peer who is initially unfamiliar with the project, the computational concepts, the type of data, and the programming language. Although it may be tempting to document all aspects of the project in a single document eventually evolving into a manuscript to be submitted for peer review, we advise not to do so for several reasons. First, it breaks the connection between the documentation and the item (e.g., code or data) being documented. Second, the level of documentation in an sFSS is likely to be much more detailed than what is provided in a typical research paper, which only outlines the main concepts and steps taken. Even the manuscript's supplementary information may not provide the same level of detail. Third, since ENCORE is used right from the start of a project, it is likely that not all parts of the project end up in a publication. For example, not all (intermediate) results become part of a publication. In addition, documenting specific approaches that failed may be equally important. Fourth, keeping the documentation modular and in the directories where it belongs helps in its maintenance. Once the manuscript will be written then the parts of the documentation that are needed can easily be incorporated. We also recommend that the overall sFSS structure, names of directories (with exceptions mentioned earlier), and names of pre-defined files are not changed; in particular for research groups that embrace this approach to achieve harmonization. Additional directories may be added if this does not effectively change the overall sFSS structure. For example, within the *Results* directory, one may create separate sub-directories for figures and tables, but one should not move *Results* to the root of the sFSS. As mentioned earlier, unused directories and files should be removed. Since an sFSS should be self-contained, any directory path used within the code or any of the pre-defined files should be relative to the top-level directory.

Results

Initial discussions and review of publications to improve reproducibility in our group for research and support projects started in 2018 (**Figure 6**), and consistently involved all group members (i.e., staff, PhD students, postdocs, and internship students). This led to the aims specified in the introduction (transparency, reproducibility, harmonization) and to a first version of ENCORE in October 2020. At the same time, we initiated a GitHub organization account (GitHub accounts, 2023) to host all ENCORE repositories from our group. The use of ENCORE 1.0 was made mandatory for all new research projects and for everyone in our group. Support projects were handled separately, as discussed below. This initial step led to the harmonization of project organization within our group, while leaving sufficient flexibility to accommodate different ways of working and organization. Currently, we have approximately 15 ENCORE research projects. In addition, we have about 50 ENCORE support projects. For education we have 8 ENCORE projects (mostly internship projects).



2018	Initial discussions about reproducibility for research and support
Oct 2020	ENCORE 1.0 GitHub organization account
Nov 2020	Evaluation. General experience →ENCORE 2.0
Jan 2021	Evaluation. Implementations → ENCORE 3.0
Sept 2022	Evaluation. Reproducibility
Nov 2022	First version of the FSS Navigator → ENCORE 3.1
April 2023	Improved the FSS Navigator Revision of documentation → ENCORE 3.5
July 2023	Revision of documentation → ENCORE 4.0
Future extensions and modifications	

Figure 6. ENCORE evolution and evaluation. ENCORE evolved through different versions incorporating changes based on practical experiences, evaluations, and group discussions. This led to gradual improvements in the ENCORE approach and documentation, broader use for research and support projects, and Git/GitHub proficiency within our research group. In turn, this led to better and more transparent organization of projects and increased reproducibility. Further changes are expected in the future.

Evolution and evaluation

Based on experience with an increasing number of projects, internal evaluations, and frequent discussions about the structure, pre-defined files, and usage guidelines, we gradually improved the initial setup. In general, over the last three years, this led to a simplification of the sFSS structure, a reduced number of pre-defined files, and better ENCORE instructions and documentation. Potential extensions and modifications are presented in the discussion section.

A first evaluation was performed one month after introducing ENCORE 1.0 to exchange experiences acquired while using it for research and support projects; This led to ENCORE 2.0. The evaluation made clear that conversion of our existing data-analysis support pipelines would take significant time and efforts to comply with the ENCORE specifications. About three months later, in January 2021, we organized a second evaluation, during which each group member presented her/his ENCORE project. This evaluation made clear that ongoing research projects adopted the sFSS structure, but that documentation (e.g., README files, lab journal) was occasionally incomplete or even missing, since keeping documentation up to data was perceived as overhead and time-consuming but also because it was not always clear what level of documentation needs to be provided. In addition, some projects did not fully adhere to the agreed upon file and directory naming and structure. This was partially due to unclear or implicit ENCORE guidelines. Most support pipelines were still not converted for various reasons (see below). The outcomes of this evaluation led to ENCORE 3.0, which comprised a further simplification of the directory structure (fewer directory levels), renaming of pre-defined files to improve consistency, and improved documentation (i.e., usage rules). In addition, the '*Sharing*' directory was added to the sFSS to share results of support projects. In September 2022, we evaluated ENCORE 3.0 to determine its status for research and support projects and, more importantly, to test if ENCORE had improved the reproducibility of our projects or not? We selected five research projects, three support projects, and one educational project. Each project was assigned to another group member, who was not involved in this project. Subsequently, each project was evaluated for the correct use of the sFSS, the pre-defined files and the level of documentation. In addition, each group member was asked to reproduce a specific subset of results from the project assigned to her/him. The most important outcome of this evaluation was that only for about half of the selected projects the results could be reproduced. Various reasons prevented (full) reproducibility such as the use of different software library versions, use of absolute directory paths that were not valid after transferring the sFSS to a different location, differences between OS's, lack of instructions to run the software, compilation problems, software errors, or difficulties in handling large datasets. In addition, some of the projects were difficult to understand due to lack of documentation (transparency) about goals and applied methodology. Most of these problems can easily be fixed, while others require more effort. Several other outcomes resulted from this evaluation. First, not being acquainted with a specific project, it sometimes was difficult to find the (core) information that is needed to reproduce and understand a project. This triggered the development of the FSS Navigator, which was implemented as an R application in ENCORE 3.1 and later re-implemented in Python and extended in ENCORE 3.5. Second, despite more than two years of discussion and joint decision about the structure of ENCORE, there was still debate about (how to use) the sFSS structure and the pre-defined files. The sFSS structure and files had been agreed upon by consensus, but still was a compromise of different structures previously used by individual group members. As a result, some parts of it seemed illogical to some members, who would have preferred a different approach. In addition, although the ENCORE user guidelines and their rationales had been discussed, agreed upon, and written down, it turned out to be difficult to memorize and apply all these rules. In our view this is also one of the main reasons that existing published guidelines are not consistently applied. At the same time, the user documentation was scattered over many files and lost consistency over the evolution of ENCORE. This prompted the changes incorporated in ENCORE 3.5 and 4.0. In specific, these are the completion of the Step-by-Step ENCORE Guide (**Supplementary File 1**) and the merger of documentation and templates directly in the README files (e.g., **Supplementary File 3**). Third, the provided level of detail for the documentation of most projects was inadequate, partially caused by unclear guidelines. For example, it was not always clear how to run the software because our guidelines didn't make explicit how this should be document and at what level of detail. One specific issue was that the lab journal often did not contain an adequate summary of (supervisory) meetings and email exchanges. It was therefore agreed upon that it is the responsibility of the project team to ensure that all relevant documentation is incorporated in the sFSS. Finally, project organization and documentation often had lower priority than doing the actual research. Moreover, since the concrete benefits of ENCORE for an individual research project are not always clearly perceived, the motivation to comply to the ENCORE guidelines was sometimes low. For these reasons, most projects did not fully adhere to the ENCORE guidelines. Nevertheless, we decided to keep the approach mandatory and to only make minor changes to the current template to prevent any delay in its further development, but also because any decision about its structure would always need to be made by consensus.

ENCORE and support projects

Support projects comprise computational/statistical analyses that are conducted as a service for researchers outside the research group. Reproducibility of support projects is equally important because of the responsibility to deliver transparent, and reproducible results while adhering to high quality standards. Moreover, the person executing a support project is also accountable in case questions or problems arise. However, support projects differ from research projects and require a slightly different approach when applying ENCORE.

In our group, support projects include the routine analysis of OMICS datasets such as bulk RNA-seq (Stark et al., 2019), adaptive immune receptor repertoire sequencing (AIRR-seq) (Robinson, 2015), and lipidomics (Vaz et al., 2015). Although each project may involve some dedicated processing, a large part of the analyses involves the application of standardized workflows that we developed over many years. Typically, these projects require much less time and effort to complete compared to research projects, which typically take many months or years to complete. Consequently, the relative overhead to fully document support projects is higher. However, since the documentation should only describe the computational procedures and not the (wet-lab/clinical) research of the supported group, the standardized workflows could be described in specific templates that can be repurposed or created automatically. Subsequently, project-specific details can be added. In addition, part of the documentation (e.g., experimental design, sample descriptions) should be delivered by the supported group and can directly be copied into the sFSS. However, the main challenge is imposed by the fact that the development of part of our data analysis workflows started pre-ENCORE and were the result of several years of development. Consequently, these support pipelines have an already established project structure and, hence, a way of working. Therefore, the transition of such workflows to ENCORE is disruptive and takes significant time and effort to complete. Due to the continuous pressure to quickly analyze the incoming stream of datasets, the analyses of these datasets have until now gotten priority over the work needed to make support fully ENCORE compatible. Newly developed pipelines for support do adhere to the ENCORE principles.

One other consideration for support projects is the use of GitHub. Since basically the same workflow is used for every new dataset, we decided not to have many (nearly) identical GitHub repositories for each individual support project. Changes in code from one project to the next are small and typically involve trivial things like changes in parameter settings, paths to datasets, etc. Moreover, since the code is always stored in the sFSS of each project, one could still determine differences between projects if necessary. For the different types of support projects, we now use different approaches. For the analysis of bulk RNAseq data, one instance of the particular data analysis pipeline is stored in GitHub that has remained unchanged since it was stored. For AIRR-seq projects, we use a single GitHub repository, but each individual project is treated as a separate branch that will never get merged into the main branch. This allows to track project-specific changes. For

Using ENCORE for projects that utilize multiple computer systems

Occasionally, one may use multiple computer systems for required computations. For example, the primary project (sFSS) location might be a personal laptop on which part of the computations are performed. Remaining computations that require, for example, High Performance Computing on dedicated computer servers or clusters should remain compatible with ENCORE. The main premise is that the sFSS remains self-containing and consistent and, consequently, that all specified directory paths in the software are relative to the root of the sFSS. This allows to copy (part of) the sFSS to another (Unix-based) computer system, perform the calculations, and then copy the sFSS including the new results back to the sFSS on the laptop.

Discussion

We presented ENCORE as a practical implementation guiding researchers on how to structure and document computational projects according to established guidelines (Brito et al., 2020; Sandve et al., 2013; Spreckelsen et al., 2020; Turkyilmaz-van der Velden et al., 2020) in order to improve transparency and reproducibility, and to allow harmonization within a research group. ENCORE does not consider replicability (sometimes referred to as repeatability), which is about strengthening scientific evidence from replication studies by other research groups using independent data, and experimental and computational methods (National Academies of Sciences & Medicine, 2019). An important facet of ENCORE is the integration of all project information (concepts, data, code, results, documentation) in a single project package that can easily be shared. Although we didn't have a pre-ENCORE baseline measurement for reproducibility in our group, ENCORE harmonized the way we work in a broad range of projects, and provided a big step forward in the organization, transparency, and reproducibility of our projects. Integration and documentation to achieve transparency have also been referred to as the third dimension of open science (Lyon, 2016) and are key to reproducibility.

ENCORE can be applied to virtually any type of computational project, is agnostic to the computational infrastructure, and can be used with any programming language or software tool the researcher wants to use. It can be applied to research, support, and education projects. Over the past few years, we have experienced that it is crucial to start using ENCORE from the start of a project and to have sufficient self-discipline to keep the project package up to date. The extensive ENCORE documentation significantly helps in using ENCORE, and is key to introducing new members of our group, who have not been involved in several years of discussion and development, to the underlying philosophy and approach.

Different areas of reproducibility are distinguished (Stodden, 2015). (i) Empirical reproducibility refers to physical experiments and the (reporting) standards associated with these. (ii) Computational reproducibility is concerned with the reproduction of results using the same data and computational methodology. (iii) Statistical reproducibility focuses on the correct use of experimental design (including sample size calculations) and statistical analyses. (iv) Ethical reproducibility refers to reporting ethics methods in biomedical research (Anderson et al., 2013). ENCORE focuses on computational reproducibility and statistical reproducibility. However, reporting standards that originate from empirical reproducibility (see below) are directly relevant for designing correct computations, while ethics reproducibility may come into play (in ENCORE) for specific (artificial intelligence) applications e.g., (Kulikowski & Maojo, 2021).

ENCORE promotes a pre-defined structure, integration of all project components, and detailed documentation and therefore has a few additional advantages and uses. For example, it may contribute to the detection of errors in the code or conceptual methodological flaws by (external) researchers, supervisor(s), and reviewers. It allows project supervisors to provide timely and more constructive feedback. Co-authors of a manuscript can more easily inspect details of the project before submission for publication to a journal to, for example, comply with the ICMJE authorship rules

(ICMJE, 2023). ENCORE also helps improving the sustainability of research projects, given the continuous flux of scientific personnel and students in academic groups. The same holds for support projects. where additional analyses are frequently requested; even after an extended period of time. Finally, we are convinced that transparency may to some extent prevent scientific misconduct, although this should not be the main driver for computational reproducibility (Diaba-Nuhoho & Amponsah-Offeh, 2021).

Software development has become an important part of computational research and, consequently, contributes to the results published in peer-reviewed journals. For this reason, it is important that the code is accessible and reliable. A recent editorial in Nature Human Behaviour (Editorial, 2021), therefore, proposed that code should be made part of the peer review process. This would require accessibility of the code (e.g., using GitHub), user documentation to install and run the code, code documentation, a software license, and one or more test datasets. It would be very time consuming and difficult to check the code itself, but reviewers could at least check if results presented in a paper could be reproduced. ENCORE enables such code peer review.

Increasing reproducibility, what is the problem?

The main hurdle to increase reproducibility of computational projects is neither the lack of guidelines nor substantial technical barriers. However, despite all discussions and initiatives concerning reproducibility, and our personal observation that many researchers agree about the importance of reproducibility, there still is a big gap to close. Regularly, during the development of ENCORE, group members brought forward various arguments for not following (ENCORE) guidelines. For example, one argument being that we are virtually never asked by peers, reviewers, or funding agencies about reproducibility. There is no direct apparent penalty for being non-reproducible, but there also is no clear reward for being reproducible. Other counter arguments to efforts to increase reproducibility are found in the paper ‘Five selfish reasons to work reproducibly’ (Markowetz, 2015), such as “I’d rather do real science than tidy up my data”. An often-heard argument concerns the amount of overhead that comes along with (ENCORE) reproducibility approaches. However, for a typical research project, the time spend on following the ENCORE approach (e.g., documentation, structuring) is negligible compared to the time spent on the actual research. On the other hand, several common arguments are often given in favour of reproducibility such as provided by Markowetz (Markowetz, 2015) and others (e.g., (Diaba-Nuhoho & Amponsah-Offeh, 2021; Sandve et al., 2013; Stodden et al., 2012). For example, reproducibility is important for trust in science, it helps writing a publication, it will save time in the long run, and it supports the sustainability of the research. Although all true, for some researchers these arguments do not seem to provide enough incentives to improve their practices. Bottom line is that reproducibility requires not only intrinsic motivation, working attitude, and discipline but also clear and direct rewards for scientists. Indeed, it is well-recognized that the way in which we reward science should be changed. For example, the Declaration on Research Assessment (DORA) is a global initiative that proposes to change the evaluation of researchers and scholarly research output by funding agencies, academic institutions, and other parties (DORA, 2023). Complementary, at the national level, similar initiatives emerged. For example, the Dutch public knowledge institutes and research funders wrote a position paper that, among others, encourages Open Science (VSNU et al., 2019). We slowly witness a change in the reward system, and we expect that this will largely contribute to reproducible science. ENCORE contributes to these efforts. A high-quality ENCORE project package can be shared through public repositories and assigned a DOI. We recently submitted a ENCORE projects to Zenodo (Van Kampen et al., 2023). An ENCORE project package containing a benchmark for spatial transcriptomics deconvolution methods will be described as an ENCORE case study in a separate publication (Mahamune et al., In preparation). We hope that such packages will, soon, be appreciated as valuable scientific output in its own right, for example, for PhD students. Another suggestion by Stodden and co-workers is to have journals and/or professional societies discern a yearly award for (young) investigators for excellent reproducible practice (Stodden et al., 2012). Both would contribute to building the researcher's scientific track record.

Limitations of ENCOREs

We consider ENCORE to be a step towards reproducible science, but it is not without several limitations and weaknesses. First, ENCORE is a compromise based on previous ways of working and, therefore, may not always fit the preferred way of working of an individual researcher. However, we believe ENCORE provides sufficient flexibility to accommodate most researchers and research practices. Second, the current FSS Navigator has limited functionality, ways to present information, and possibilities to configuration options. Third, a shortcoming of ENCORE is the lack of explicit links between results, code, data, and concepts other than those imposed by the sFSS structure, the paths in the code, and/or the documentation. In particular, the documentation has an important function in gluing the project parts together. Consequently, identifying the relationships between these items is not impossible, but requires some effort from the person using a project package. Improved and explicit linking approaches would further improve transparency and reproducibility. A possible approach is the use of YAML (YAML, 2023) as demonstrated by Spreckelsen and co-workers (Spreckelsen et al., 2020). Fourth, another challenge that is only partially addressed by ENCORE concerns the preservation of the full computing environment. This environment is defined by (interdependencies of) the operating system, software tools, versions and dependencies, programming language libraries, etc. Gruning and co-workers proposed a software stack of interconnected technologies to preserve the computing environment (Gruning, Chilton, et al., 2018). This stack comprises (i) (Bio)Conda (Anaconda Software Distribution, 2020; Gruning, Dale, et al., 2018) to provide virtual execution environments addressing software versions and dependencies, (ii) container platforms such as Docker (Nust et al., 2020) to preserve other aspects of the runtime environment, and (iii) virtual machines using cloud systems or dedicated applications such as VMware, to overcome the dependencies on the operating system and hardware. We are currently investigating how to best approach this within the ENCORE environment. Reproducibility can further be improved by using scientific workflow systems, which are developed to modularize and execute steps in computations. Many workflow systems are available, including Galaxy (Galaxy, 2022), KNIME (Berthold et al., 2008), Snakemake (Molder et al., 2021) and Nextflow (Di Tommaso et al., 2017). These workflow systems improve reproducibility and help to maintain and share computational analyses. They also allow incorporation of steps that otherwise would have been performed manually. Our group has used workflow systems in the past (e.g., (Boekel et al., 2015; Madougou et al., 2012)) but did not make it part of the ENCORE approach at this moment, as we believe this might be too disruptive for some researchers. Yet, we encourage our group members to use a workflow system of choice. Fifth, ENCORE enables version control of software code and documentation but doesn't have an explicit approach towards version control of concepts, data, and results, which might be equally important in some cases for provenance. Currently, one would have to store different versions within the sFSS and provide the corresponding documentation. Sixth, ENCORE enables the sharing of complete projects. However, there may be information in the project package that one doesn't want to share (e.g., new research ideas in the lab journal) or that one isn't allowed to share (e.g., patient data, controlled access data obtained from public repositories, or PDF copies of copyright protected scientific publications). Currently, ENCORE does not comprise an automatic mechanism to remove such information. However, we are investigating the possibility of developing an application to perform such a task based on a configuration file (.FSSignore) in which one can specify (parts of) files and subdirectories that should not be shared.

Training, scientific computing, and software engineering

Over the past two decades, an increasing number of biomedical researchers have become involved in computational research. Many of these researchers have never been formally trained in scientific computing and software engineering (e.g., design, programming, documentation) (Hermann & Fehr, 2022; Martin, 2008), software version control (Blischak et al., 2016; Perez-Riverol et al., 2016), the use of high-performance computing infrastructures, the use of Unix/Linux which is still the major platform for scientific computing, algorithm design, the use of (Jupyter, R) notebooks (Rule et al., 2019), etc. This even holds for part of the researchers working in computational groups. Lack of such skills may negatively affect reliability of software and, consequently, computational reproducibility. For example, software may be poorly designed and documented, making it difficult to understand and use. Software engineering is a discipline in its own and includes the design, implementation, documentation, testing and deployment of software. Following best practices for scripting, functional programming, or objective-oriented programming may significantly improve the quality of the code but requires training and experience. In addition, software documentation occasionally leaves much to desire. In a recent report, it was concluded that researchers are generally not aware for whom they write documentation and what documentation is required (Hermann & Fehr, 2022). Currently, ENCORE does not provide specific instructions for coding style and documentation design, because it is probably more effective to train scientist in the art of software engineering. Awareness of tools that help to (automatically) generate documentation such as Sphinx (Brandl, 2021) for Python, and r2readthedocs (r2readthedocs, 2023) and roxygen2 (Roxygen2, 2023) for R, will also help to improve reproducibility. We used Sphinx for the documentation of the FSS Navigator (Van Kampen et al., 2023), Another useful resource is the software management plan developed by the Netherlands eScience Center and the Dutch Research Council (NWO) (Martinez-Ortiz et al., 2023). In general, training on reproducibility approaches could already significantly improve the current situation and will at least create awareness of the tremendous amount of literature about many aspects of sound scientific computing practices (Carey & Papin, 2018; Lapp et al., 2022; Larcombe et al., 2017; Toelch & Ostwald, 2018; Wilson et al., 2014; Wilson et al., 2017). In addition, senior researchers should strongly promote reproducibility and support, explain, and instruct junior researchers.

Complementary reporting guidelines and standards

To facilitate and improve reproducibility throughout the complete research lifecycle (**Figure 1**), numerous guidelines, policies, and standards have been developed (FAIRsharing.org, 2023) to guide and facilitate the detailed documentation of the steps á this process are lacking. Many reporting guidelines provide structured tools specifying the minimum information required for specific study types and largely contribute to the transparency, understanding, and reproducibility of a study (EQUATOR Network, 2023). Examples include guidelines for clinical trials (CONSORT) (Schulz et al., 2010), diagnostic accuracy studies (STARD) (Bossuyt et al., 2015), and observational studies (STROBE) (von Elm et al., 2007). Virtually all of these minimum information standards and reporting guidelines require the specification of statistical and computational methods that were used in a study. However, exact requirements to specify such methods are often lacking. In addition, an increasing number of scientific journals have their own guidelines. One example is the Nature Reporting Summary (Nature, 2023b) that partially relies on existing reporting guidelines and FAIR. Nature also has a specific software and algorithm policy with requirements about availability (e.g., using GitHub (Blischak et al., 2016) or Zenodo), use of an open-source license (Open Source Initiative, 2023), and code review (Nature, 2023a). This policy requires providing a complete and detailed description of the code functionality. The PLOS Computational Biology journal requires that editors and reviewers can access the software, reproduce the results, and run the software on a deposited dataset with provided control parameters (PLOS Computational Biology, 2023). The Science journal TOP guidelines require data and computer code to be available (Science, 2023). Interestingly, a study published in 2018 showed that despite these guidelines, many computational studies were not reproducible (Stodden et al., 2018). Clearly, computational projects require their own specific guidelines and standards to guarantee transparency and reproducibility. This was also recognized by the artificial intelligence community, which started initiatives to develop specific AI-oriented guidelines (Haibe-Kains et al., 2020; Ibrahim, Liu, & Denniston, 2021; Ibrahim, Liu, et al., 2021). To the best of our knowledge, there are not many (practical) reporting guidelines nor standards available for computational research that are routinely used in practice. Nevertheless, there are various initiatives to improve this situation. For example, recently detailed guidelines for a standardized file system structure for scientific data were published by Spreckelsen and co-workers (Spreckelsen et al., 2020), which inspired the sFSS used in ENCORE. However, they use a different organization, i.e., the top-level of their file system layout is not an individual project like in ENCORE but an experiment, simulation, data analysis, or publication. Recently, it has been suggested to apply the FAIR principles to software (Barker et al., 2022; Fair Software, 2023). Other examples include, the ICERM implementation and archiving criteria for software (Stodden et al., 2012), the Adaptive Immune Receptor Repertoire (AIRR) software guidelines (AIRR, 2023), the Software Ontology to describe software used to store, manage and analyze data (Malone et al., 2014), and the EDAM ontology to describe bioinformatics operations (Ison et al., 2013). For simulation-based research there are initiatives like the Minimum Information About a Simulation Experiment (MIASE) guidelines (Waltemath et al., 2011), the corresponding simulation experiment description markup language (SED-ML) (Smith et al., 2021), COMBINE/OMEX to share and reproduce modeling projects (Bergmann et al., 2014), and a range of others (Tatka et al., 2022). For the further development of ENCORE, we will need to consider which of these standards are relevant and how to incorporate them in the ENCORE approach.

Acknowledgements

(Former) members of the Bioinformatics Laboratory (www.bioinformaticslaboratory.eu) at the Amsterdam University Medical Centers are acknowledged for their contributions and ideas to develop the ENCORE approach. This work is supported by COSMIC which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 765158, by ARCAID (https://www.arcaid-h2020.eu), which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 847551, by CLOSIT, which is a collaboration project financed by the Ministry of Economic Affairs by means of the PPP Allowance made available by the Top Sector Life Sciences and Health.

References

AIRR. (2023). *AIRR Software Guidelines; Adaptive Immune Receptor Repertoire Software Guidelines*. Retrieved March 2023 from DOI: 10.25504/FAIRsharing.eNSzPf

Anaconda Software Distribution. (2020). *Anaconda Documentation. Anaconda Inc.* . Retrieved July 2023 from https://docs.anaconda.com/

Anderson, J. A., Eijkholt, M., & Illes, J. (2013). Ethical reproducibility: towards transparent reporting in biomedical research. *Nat Methods*, *10*(9), 843-845. https://doi.org/10.1038/nmeth.2564

Barker, M., Chue Hong, N. P., Katz, D. S., Lamprecht, A. L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., & Honeyman, T. (2022). Introducing the FAIR Principles for research software. *Sci Data*, *9*(1), 622. https://doi.org/10.1038/s41597-022-01710-x

Bergmann, F. T., Adams, R., Moodie, S., Cooper, J., Glont, M., Golebiewski, M., Hucka, M., Laibe, C., Miller, A. K., Nickerson, D. P., Olivier, B. G., Rodriguez, N., Sauro, H. M., Scharm, M., Soiland-Reyes, S., Waltemath, D., Yvon, F., & Le Novere, N. (2014). COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics*, *15*(1), 369. https://doi.org/10.1186/s12859-014-0369-z

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., & Wiswedel, B. (2008). KNIME: The Konstanz Information Miner. Data Analysis, Machine Learning and Applications, Berlin, Heidelberg.

Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). A Quick Introduction to Version Control with Git and GitHub. *PLoS Comput Biol*, *12*(1), e1004668. https://doi.org/10.1371/journal.pcbi.1004668

Boekel, J., Chilton, J. M., Cooke, I. R., Horvatovich, P. L., Jagtap, P. D., Kall, L., Lehtio, J., Lukasse, P., Moerland, P. D., & Griffin, T. J. (2015). Multi-omic data analysis using Galaxy. *Nat Biotechnol*, *33*(2), 137-139. https://doi.org/10.1038/nbt.3134

Bossuyt, P. M., Reitsma, J. B., Bruns, D. E., Gatsonis, C. A., Glasziou, P. P., Irwig, L., Lijmer, J. G., Moher, D., Rennie, D., de Vet, H. C., Kressel, H. Y., Rifai, N., Golub, R. M., Altman, D. G., Hooft, L., Korevaar, D. A., Cohen, J. F., & Group, S. (2015). STARD 2015: an updated list of essential items for reporting diagnostic accuracy studies. *BMJ*, *351*, h5527. https://doi.org/10.1136/bmj.h5527

Brandl, G. (2021). *Sphinx documentation*. Retrieved August 2023 from https://www.sphinx-doc.org/en/master/

Brito, J. J., Li, J., Moore, J. H., Greene, C. S., Nogoy, N. A., Garmire, L. X., & Mangul, S. (2020). Recommendations to enhance rigor and reproducibility in biomedical research. *Gigascience*, *9*(6). https://doi.org/10.1093/gigascience/giaa056

Carey, M. A., & Papin, J. A. (2018). Ten simple rules for biologists learning to program. *PLoS Comput Biol*, *14*(1), e1005871. https://doi.org/10.1371/journal.pcbi.1005871

CERN, & OpenAIRE. (2023). *Zenodo*. CERN. Retrieved March 2023 from https://www.zenodo.org/

Deardorff, A. (2020). Assessing the impact of introductory programming workshops on the computational reproducibility of biomedical workflows. *PLoS One*, *15*(7), e0230697. https://doi.org/10.1371/journal.pone.0230697

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nat Biotechnol*, *35*(4), 316-319. https://doi.org/10.1038/nbt.3820

Diaba-Nuhoho, P., & Amponsah-Offeh, M. (2021). Reproducibility and research integrity: the role of scientists and institutions. *BMC Res Notes*, *14*(1), 451. https://doi.org/10.1186/s13104-021-05875-3

DORA. (2023). *The Declaration on Research Assessment*. Retrieved July 2023 from https://sfdora.org/

Editorial. (2021). Supporting computational reproducibility through code review. *Nat Hum Behav*, *5*(8), 965-966. https://doi.org/10.1038/s41562-021-01190-w

EQUATOR Network. (2023). *What is a reporting guideline?* . Retrieved March 2023 from https://www.equator-network.org/about-us/what-is-a-reporting-guideline/

Fair Software. (2023). *Five Recommendations for FAIR Software*. Retrieved March 2023 from https://fair-software.eu/

FAIRsharing.org. (2023). *A curated, informative and educational resource on data and metadata standards, inter-related to databases and data policies*. Retrieved March 2023 from https://fairsharing.org/

Fidler, F., & Wilcox, J. (2021). *Reproducibility of Scientific Results*. Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2021/entries/scientific-reproducibility

Figshare. (2023). *Repository to make research outputs available in a citable, shareable and discoverable manner*. Retrieved March 2023 from https://figshare.com/

Galaxy, C. (2022). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. *Nucleic Acids Res*, *50*(W1), W345-W351. https://doi.org/10.1093/nar/gkac247

Garijo, D., Menager, H., Hwang, L., Trisovic, A., Hucka, M., Morrell, T., Allen, A., Task Force on Best Practices for Software, R., & SciCodes, C. (2022). Nine best practices for research software registries and repositories. *PeerJ Comput Sci*, *8*, e1023. https://doi.org/10.7717/peerj-cs.1023

GitHub. (2023). *Types of GitHub accounts*. Retrieved 28 July 2023 from https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts

GitHub accounts. (2023). *Overview of different types of GitHub accounts*. Retrieved April 2023 from https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts

Gruning, B., Chilton, J., Koster, J., Dale, R., Soranzo, N., van den Beek, M., Goecks, J., Backofen, R., Nekrutenko, A., & Taylor, J. (2018). Practical Computational Reproducibility in the Life Sciences. *Cell Syst*, *6*(6), 631-635. https://doi.org/10.1016/j.cels.2018.03.014

Gruning, B., Dale, R., Sjodin, A., Chapman, B. A., Rowe, J., Tomkins-Tinch, C. H., Valieris, R., Koster, J., & Bioconda, T. (2018). Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods*, *15*(7), 475-476. https://doi.org/10.1038/s41592-018-0046-7

Haibe-Kains, B., Adam, G. A., Hosny, A., Khodakarami, F., Massive Analysis Quality Control Society Board of, D., Waldron, L., Wang, B., McIntosh, C., Goldenberg, A., Kundaje, A., Greene, C. S., Broderick, T., Hoffman, M. M., Leek, J. T., Korthauer, K., Huber, W., Brazma, A., Pineau, J., Tibshirani, R., . . . Aerts, H. (2020). Transparency and reproducibility in artificial intelligence. *Nature*, *586*(7829), E14-E16. https://doi.org/10.1038/s41586-020-2766-y

Hermann, S., & Fehr, J. (2022). Documenting research software in engineering science. *Sci Rep*, *12*(1), 6567. https://doi.org/10.1038/s41598-022-10376-9

Hunter-Zinck, H., de Siqueira, A. F., Vasquez, V. N., Barnes, R., & Martinez, C. C. (2021). Ten simple rules on writing clean and reliable open-source scientific software. *PLoS Comput Biol*, *17*(11), e1009481. https://doi.org/10.1371/journal.pcbi.1009481

Ibrahim, H., Liu, X., & Denniston, A. K. (2021). Reporting guidelines for artificial intelligence in healthcare research. *Clin Exp Ophthalmol*, *49*(5), 470-476. https://doi.org/10.1111/ceo.13943

Ibrahim, H., Liu, X., Rivera, S. C., Moher, D., Chan, A. W., Sydes, M. R., Calvert, M. J., & Denniston, A. K. (2021). Reporting guidelines for clinical trials of artificial intelligence interventions: the SPIRIT-AI and CONSORT-AI guidelines. *Trials*, *22*(1), 11. https://doi.org/10.1186/s13063-020-04951-6

ICMJE. (2023). *International committee of Medical Journal Editors. Defining the Role of Authors and Contributors*. Retrieved March 2023 from https://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Med*, *2*(8), e124. https://doi.org/10.1371/journal.pmed.0020124

Ison, J., Kalas, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., Malone, J., Lopez, R., Pettifer, S., & Rice, P. (2013). EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, *29*(10), 1325-1332. https://doi.org/10.1093/bioinformatics/btt113

Karimzadeh, M., & Hoffman, M. M. (2018). Top considerations for creating bioinformatics software documentation. *Brief Bioinform*, *19*(4), 693-699. https://doi.org/10.1093/bib/bbw134

Kulikowski, C., & Maojo, V. M. (2021). COVID-19 pandemic and artificial intelligence: challenges of ethical bias and trustworthy reliable reproducibility? *BMJ Health Care Inform*, *28*(1). https://doi.org/10.1136/bmjhci-2021-100438

Lapp, Z., Sovacool, K. L., Lesniak, N., King, D., Barnier, C., Flickinger, M., Kruger, J., Armour, C. R., Lapp, M. M., Tallant, J., Diao, R., Oneka, M., Tomkovich, S., Anderson, J. M., Lucas, S. K., & Schloss, P. D. (2022). Developing and deploying an integrated workshop curriculum teaching computational skills for reproducible research. *J Open Source Educ*, *5*(47). https://doi.org/10.21105/jose.00144

Larcombe, L., Hendricusdottir, R., Attwood, T. K., Bacall, F., Beard, N., Bellis, L. J., Dunn, W. B., Hancock, J. M., Nenadic, A., Orengo, C., Overduin, B., Sansone, S. A., Thurston, M., Viant, M. R., Winder, C. L., Goble, C. A., Ponting, C. P., & Rustici, G. (2017). ELIXIR-UK role in bioinformatics training at the national level and across ELIXIR. *F1000Res*, *6*. https://doi.org/10.12688/f1000research.11837.1

Lashgari, D., Merino Tejero, E., Meyer-Hermann, M., Claireaux, M. A. F., van Gils, M. J., Hoefsloot, H. C. J., & van Kampen, A. H. C. (2022). From affinity selection to kinetic selection in Germinal Centre modelling. *PLoS Comput Biol*, *18*(6), e1010168. https://doi.org/10.1371/journal.pcbi.1010168

Lyon, L. (2016). Transparency: The Emerging Third Dimension of Open Science and Open Data. *Liber Quarterly*, *25*(4), 153-171. https://doi.org/10.18352/lq.10113

Madougou, S., Santcroos, M., Benabdelkader, A., van Schaik, B. D. C., Shahand, S., Korkhov, V., van Kampen, A. H. C., & Olabarriaga, S. D. (2012). Provenance for distributed biomedical workflow execution. *Stud*

Health Technol Inform, *175*, 91-100.

Mahamune, U., Jongejan, A., Moerland, P. D., & Van Kampen, A. H. C. (In preparation). ENCORE. A reproducibility case study for benchmarking deconvolution methods for spatial transcriptomics.

Malone, J., Brown, A., Lister, A. L., Ison, J., Hull, D., Parkinson, H., & Stevens, R. (2014). The Software Ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation. *J Biomed Semantics*, *5*, 25. https://doi.org/10.1186/2041-1480-5-25

Margan, D., & Čandrlić, S. (2015). *The success of open source software: A review* 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),

Markdown. (2023). *Everything you need to learn Markdown*. Retrieved 28 July 2023 from https://www.markdownguide.org/about/

Markowetz, F. (2015). Five selfish reasons to work reproducibly. *Genome Biol*, *16*, 274. https://doi.org/10.1186/s13059-015-0850-7

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson.

Martinez-Ortiz, C., Martinez Lavanchy, P., Sesink, L., Olivier, B. G., Meakin, J., de Jong, M., & Cruz, M. (2023). *Practical guide to Software Management Plans (1.1)*. https://zenodo.org/record/7589725

Mendes, P. (2018). Reproducible Research Using Biomodels. *Bull Math Biol*, *80*(12), 3081-3087. https://doi.org/10.1007/s11538-018-0498-z

Merino Tejero, E., Lashgari, D., Garcia-Valiente, R., Gao, X., Crauste, F., Robert, P. A., Meyer-Hermann, M., Martinez, M. R., van Ham, S. M., Guikema, J. E. J., Hoefsloot, H., & van Kampen, A. H. C. (2020). Multiscale Modeling of Germinal Center Recapitulates the Temporal Transition From Memory B Cells to Plasma Cells Differentiation as Regulated by Antigen Affinity-Based Tfh Cell Help. *Front Immunol*, *11*, 620716. https://doi.org/10.3389/fimmu.2020.620716

Molder, F., Jablonski, K. P., Letcher, B., Hall, M. B., Tomkins-Tinch, C. H., Sochat, V., Forster, J., Lee, S., Twardziok, S. O., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., & Koster, J. (2021). Sustainable data analysis with Snakemake. *F1000Res*, *10*, 33. https://doi.org/10.12688/f1000research.29032.2

National Academies of Sciences, E., & Medicine. (2019). *Reproducibility and Replicability in Science*. The National Academies Press. https://doi.org/doi:10.17226/25303

Nature. (2023a). *Availability and peer review of computer code and algorithm*. Retrieved March 2023 from https://www.nature.com/nature-portfolio/editorial-policies/reporting-standards#availability-of-computer-code

Nature. (2023b). *Reporting Summary,*. Retrieved March 2023 from https://www.nature.com/documents/nr-reporting-summary-flat.pdf

Nust, D., Sochat, V., Marwick, B., Eglen, S. J., Head, T., Hirst, T., & Evans, B. D. (2020). Ten simple rules for writing Dockerfiles for reproducible data science. *PLoS Comput Biol*, *16*(11), e1008316. https://doi.org/10.1371/journal.pcbi.1008316

Open Source Initiative. (2023). *OSI Approved Licenses*. Retrieved March 2023 from https://opensource.org/licenses/

Papin, J. A., Mac Gabhann, F., Sauro, H. M., Nickerson, D., & Rampadarath, A. (2020). Improving reproducibility in computational biology research. *PLoS Comput Biol*, *16*(5), e1007881. https://doi.org/10.1371/journal.pcbi.1007881

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost Fda, V., Fufezan, C., Ternent, T., Eglen, S. J., Katz, D. S., Pollard, T. J., Konovalov, A., Flight, R. M., Blin, K., & Vizcaino, J. A. (2016). Ten Simple Rules for Taking Advantage of Git and GitHub. *PLoS Comput Biol*, *12*(7), e1004947. https://doi.org/10.1371/journal.pcbi.1004947

PLOS Computational Biology. (2023). *Material, Software, and Code Sharing*. Retrieved March 2023 from https://journals.plos.org/ploscompbiol/s/materials-software-and-code-sharing

Pulverer, B. (2018). Open Access-or Open Science? *EMBO J*, *37*(24). https://doi.org/10.15252/embj.2018101215

r2readthedocs. (2023). *Convert R package documentation to a 'readthedocs' website.* Retrieved August 2023 from https://github.com/ropenscilabs/r2readthedocs/

Ram, K. (2013). Git can facilitate greater reproducibility and increased transparency in science. *Source Code Biol Med*, *8*(1), 7. https://doi.org/10.1186/1751-0473-8-7

Rigden, D. J., & Fernandez, X. M. (2023). The 2023 Nucleic Acids Research Database Issue and the online molecular biology database collection. *Nucleic Acids Res*, *51*(D1), D1-D8. https://doi.org/10.1093/nar/gkac1186

Robinson, W. H. (2015). Sequencing the functional antibody repertoire--diagnostic and therapeutic discovery. *Nat Rev Rheumatol*, *11*(3), 171-182. https://doi.org/10.1038/nrrheum.2014.220

Roxygen2. (2023). *Dynamic documentation system*. Retrieved August 2023 from https://cran.r-project.org/web/packages/roxygen2/index.html

Rule, A., Birmingham, A., Zuniga, C., Altintas, I., Huang, S. C., Knight, R., Moshiri, N., Nguyen, M. H., Rosenthal, S. B., Perez, F., & Rose, P. W. (2019). Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLoS Comput Biol*, *15*(7), e1007007. https://doi.org/10.1371/journal.pcbi.1007007

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLoS Comput Biol*, *9*(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Schulz, K. F., Altman, D. G., Moher, D., & Group, C. (2010). CONSORT 2010 statement: updated guidelines for reporting parallel group randomized trials. *Ann Intern Med*, *152*(11), 726-732. https://doi.org/10.7326/0003-4819-152-11-201006010-00232

Science. (2023). *Research Standards. Transparency and Openness Promotion (TOP) guidelines/*. Retrieved March 2023 from https://www.science.org/content/page/science-journals-editorial-policies#TOP-guidelines

Smith, L. P., Bergmann, F. T., Garny, A., Helikar, T., Karr, J., Nickerson, D., Sauro, H., Waltemath, D., & Konig, M. (2021). The simulation experiment description markup language (SED-ML): language specification for level 1 version 4. *J Integr Bioinform*, *18*(3), 20210021. https://doi.org/10.1515/jib-2021-0021

Spreckelsen, F., Rüchardt, B., Lebert, J., Luther, S., Parlitz, U., & Schlemmer, A. (2020). Guidelines for a Standardized Filesystem Layout for Scientific Data. *Data*, *5*(2). https://doi.org/10.3390/data5020043

Stark, R., Grzelak, M., & Hadfield, J. (2019). RNA sequencing: the teenage years. *Nat Rev Genet*, *20*(11), 631-656. https://doi.org/10.1038/s41576-019-0150-2

Stodden, V. (2015). Reproducing Statistical Results. *Annual Review of Statistics and Its Application*, *2*(1), 1-19. https://doi.org/10.1146/annurev-statistics-010814-020127

Stodden, V., Bailey, D. H., Borwein, J., LeVeque, R. J., Rider, W., & Stein, W. (2012). Setting the Default to Reproducible Reproducibility in Computational and Experimental Mathematics. *ICERM workshop*. https://icerm.brown.edu/topical_workshops/tw12-5-rcem/icerm_report.pdf

Stodden, V., Seiler, J., & Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proc Natl Acad Sci U S A*, *115*(11), 2584-2589. https://doi.org/10.1073/pnas.1708290115

Stupple, A., Singerman, D., & Celi, L. A. (2019). The reproducibility crisis in the age of digital medicine. *NPJ Digit Med*, *2*, 2. https://doi.org/10.1038/s41746-019-0079-z

Tatka, L. T., Smith, L. P., Hellerstein, J. L., & Sauro, H. M. (2022). Adapting Modeling and Simulation Credibility Standards to Computational Systems Biology. *arXiv*. https://doi.org/https://doi.org/10.48550/arXiv.2301.06007

Tiwari, K., Kananathan, S., Roberts, M. G., Meyer, J. P., Sharif Shohan, M. U., Xavier, A., Maire, M., Zyoud, A., Men, J., Ng, S., Nguyen, T. V. N., Glont, M., Hermjakob, H., & Malik-Sheriff, R. S. (2021). Reproducibility in systems biology modelling. *Mol Syst Biol*, *17*(2), e9982. https://doi.org/10.15252/msb.20209982

Toelch, U., & Ostwald, D. (2018). Digital open science-Teaching digital tools for reproducible and transparent research. *PLoS Biol*, *16*(7), e2006022. https://doi.org/10.1371/journal.pbio.2006022

Turkyilmaz-van der Velden, Y., Dintzner, N., & Teperek, M. (2020). Reproducibility Starts from You Today. *Patterns (N Y)*, *1*(6), 100099. https://doi.org/10.1016/j.patter.2020.100099

van der Heyden, M. A. G., & van Veen, T. A. B. (2018). Gold open access: the best of both worlds. *Neth Heart J*, *26*(1), 3-4. https://doi.org/10.1007/s12471-017-1064-2

Van Kampen, A. H. C., Jongejan, A., & Mahamune, U. (2023). *The standardized file system structure (FSS) navigator repository*. Retrieved May 2023 from https://github.com/EDS-Bioinformatics-Laboratory/FSS-Navigator

Vaz, F. M., Pras-Raves, M., Bootsma, A. H., & van Kampen, A. H. (2015). Principles and practice of lipidomics. *J Inherit Metab Dis*, *38*(1), 41-52. https://doi.org/10.1007/s10545-014-9792-6

von Elm, E., Altman, D. G., Egger, M., Pocock, S. J., Gotzsche, P. C., Vandenbroucke, J. P., & Initiative, S. (2007). The Strengthening the Reporting of Observational Studies in Epidemiology (STROBE) statement: guidelines for reporting observational studies. *Ann Intern Med*, *147*(8), 573-577. https://doi.org/10.7326/0003-4819-147-8-200710160-00010

VSNU, NFU, KNAW, NWO, & ZonMw. (2019). *Room for everyone's talent: towards a new balance in the recognition and rewards for academics*. https://recognitionrewards.nl/wp-content/uploads/2020/12/position-paper-room-for-everyones-talent.pdf

Waltemath, D., Adams, R., Beard, D. A., Bergmann, F. T., Bhalla, U. S., Britten, R., Chelliah, V., Cooling, M. T., Cooper, J., Crampin, E. J., Garny, A., Hoops, S., Hucka, M., Hunter, P., Klipp, E., Laibe, C., Miller, A. K., Moraru, I., Nickerson, D., . . . Le Novere, N. (2011). Minimum Information About a Simulation Experiment (MIASE). *PLoS Comput Biol*, *7*(4), e1001122. https://doi.org/10.1371/journal.pcbi.1001122

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., . . . Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data*, *3*, 160018. https://doi.org/10.1038/sdata.2016.18

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., & Wilson, P. (2014). Best practices for scientific computing. *PLoS Biol*, *12*(1), e1001745. https://doi.org/10.1371/journal.pbio.1001745

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLoS Comput Biol*, *13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

YAML. (2023). *YAML Ain't Markup Language*. Retrieved August 2023 from https://yaml.org/

Ziemann, M., Poulain, P., & Bora, A. (2023). *The five pillars of computational reproducibility: Bioinformatics and beyond*. https://doi.org/10.31219/osf.io/4pd9n