LabInform: A Modular Laboratory Information System Built From Open Source Components

Till Biskup[1, *]

[1] *Institut für Physikalische Chemie, Albert-Ludwigs-Universität Freiburg, Albertstr. 21, 79104 Freiburg, Germany*

A framework for reproducible data analysis is only half the battle if it comes to reproducible research. Additional essential requirements are a way to safely store both, raw data and metadata and a method to uniquely refer to a dataset or any piece of information. Such unique identifier is fully independent of the actual place the information referred to is stored and does not change over time. Additionally, numeric IDs for samples and alike come in quite handy. A knowledge base and an electronic lab notebook, both based on wiki software and thus easily accessible requiring only a web browser and connection to the intranet, complete the system. Overarching design rules are simplicity, robustness and sustainability, focussing on small-scale deployment of the system retaining compatibility with future developments and community efforts. Key aspects in setting up the system are its use of well-proven open-source tools combined with maximal modularity, resulting in a low entry threshold and allowing to implement and develop it along the way of focussing on actual research.

I. INTRODUCTION

The time scale of science and research exceeds by far the average time a scientist spends in academia. On the other hand, science is deeply rooted in both, the promise to rely on a ground as solid as possible and the confidence in the scientists acquiring scientific knowledge to perform their research in a reliable and reproducible way [1]. To be relevant for generating scientific knowledge that lasts or at least forms a solid ground for those building on top of it, researchers are therefore responsible to take any effort necessary to ensure reproducible research [2, 3]. This requires to establish and use an infrastructure allowing to collect and reassess previous results even long after those originally participating in their generation have left the field [4, 5].

Now one might say that exactly this is the realm of scientific publications. However, with the advent of computers and the possibility to perform ever more complex experiments and generate unprecedented amounts of data that no single person can ever oversee, this seems no longer adequate. Additionally, software used for analysing the data underlying the conclusions presented in scientific publications are rarely published along with the results [6, 7]. As a matter of fact, an average spectroscopist can nowadays acquire more data within a year than she or he can properly analyse within the next decade. Two aspects normally limit the analysis of the data collected: Lack of appropriate software and strategies capable of coping with the inherent complexity of the task [8–11], and missing access to the data and accompanying metadata, *i.e.*, information about the (numeric) data in a structured and useful way.

Many scientific disciplines have realised these problems and have developed strategies to cope with them. A good example are widely recognised and adopted systems for data deposition, a prime example being the Protein Data Bank (PDB) for crystal structures of proteins [12, 13] dating back to the 1970s [14]. On the other hand, many national and transnational research agencies nowadays routinely require their applicants to outline data management plans together with their research proposals [5].

The other side of the coin seems to be the personal experience of the author. There is no generally recognised system to store data in his particular field of research (magnetic resonance and in particular electron paramagnetic resonance spectroscopy) he would be aware of. It seems rather that it is in the realm of every single scientist to come up with a solution how to store data in a sustainable and accessible way for periods well exceeding the presence of an average PhD student who originally recorded them. Admittedly, the task is in no way simple, requiring time to develop ideas for solutions and appropriate skills, mostly in software engineering and IT management, not regularly found in a scientific academic context with small groups and mostly individual researchers. Nevertheless, neither the problem nor strategies how to tackle it are in any way new. All that is missing might be a broader recognition of the problem and its consequences for reproducible research as well as simple yet powerful systems every scientist with average training in using computer systems and software development can implement and extend in context of the own research.

We present here strategies for a laboratory information system, termed LabInform, taking care of the pressing need for reliable and reproducible research, namely long-term archiving of research data and accompanying metadata in a safe, secure and accessible way. Combined with a framework for data analysis that ensures a gapless history of each data processing step to be written, this will pave the way towards more reliable and reproducible research with a rather minimal effort. A data analysis framework might be based on ideas implemented in the ASpecD framework [15] developed by the author. The focus of the concepts and strategies presented here is on a highly modular system built from well-proven open

* E-mail: research@till-biskup.de

source components. To possibly be successful, such a system needs to be user-friendly, robust, easy to incorporate into existing infrastructure and workflows and without need for any central facilities or increased administrative effort.

II. INTENDED TARGET AUDIENCE AND PREREQUISITES

As mentioned, different scientific communities not only have different needs, but as well a different level of organisation and availability of widely adopted systems for long-term archival of data in an accessible way. The ideas presented here are mainly targeted at spectroscopists that lack the established infrastructure, but are interested in setting up a local system on their own in their research group. The promise is simple: Having a small-scale system is better than no system, and if eventually a larger solution emerges, converting between formats should be straightforward, as long as both, source and target are open-source and well documented.

Given the generally rather limited access to funds in science, many researchers will not be willing to spend money on systems they cannot rate in terms of longevity, availability and fitness for the intended purpose. Hence, a system built from well-proven open source software components provides a big advantage. Next comes the need to be platform-independent, given the largely heterogeneous hardware and operating system landscape in science. Again connected to the available funding and man power, a laboratory information system should not impose the need for a centrally managed infrastructure and require only minimal effort for administration during routine operations. This all boils down to a highly modular system composed of independent modules that integrate well with each other. To quote a phrase from the world of agile software development: 'Think big, start small, scale fast'.

III. DESIGN STRATEGIES

The development of LabInform followed three simple yet powerful design strategies that are shortly detailed. Understanding these strategies and why they are essential is key to both, using and extending the system.

A. Simple

Science in itself is an utterly complex undertaking, hence the scientist should focus as much as possible on the inherent scientific tasks and not be busy with keeping alive a system for reproducible research. To quote Alfred North Whitehead: 'Civilization advances by extending the number of important operations we can perform without thinking.' [16] LabInform consists of a small set of simple rules or best practices that are easy to grasp and easy to implement. Simplicity, however, does not imply minor impact. On the contrary, as simple things are often easier to implement, it helps the system to get employed in real-world tasks.

B. Robust

All the information contained in the system should be accessible without having to use any of the rules imposed by the system or any particular piece of software, except of the very basic tools of the underlying operating system. A simple example: All information should be stored in text files within the file system, in a sensible hierarchy of directories with well-chosen and easily comprehensible names. The only exception to using text files is numeric data that may be stored as IEEE 754 binaries [17]. And of course, original raw data, in whatever proprietary vendor format they may appear, should be archived as well, together with an export to a more accessible format.

The reason for these rules is simple: Nobody can foresee how long a certain software will exist, be maintained, or even only be executable. The time frame of scientific research is but far longer than the average life time of a computer program or system. What we can basically rely on is access to even ancient file systems, and copying files byte-wise from one storage medium to another can (and should) be done entirely automatic. The success of the Unix operating system and its descendants relies to a good part on those simple yet powerful decisions [18].

Databases and other tools greatly facilitating day-by-day handing of both, the system and the information contained, shall and will be implemented and used. However, the system is not allowed to rely on them, nor even to be aware of this additional outer layer of complexity [19]. This is similar to using an integrated development environment (IDE) for software development. While extremely powerful and greatly enhancing development speed if used wisely, every sensible programmer should both, know about the underlying processes that are masked by a shiny user interface, and know how to do every of these tasks entirely by hand [20].

C. Sustainable

As mentioned in the last section, the time frame of science and software development are largely divergent. Software is usually quite short-lived, whereas ideas transcendent a concrete implementation and tend to be very longstanding. None of the ideas presented here is particularly new, nor is their application in a research context [21]. Nevertheless, own experience shows that many scientists reinvent the wheel over and over again, the author being no exception.

The LabInform system is designed with the single researcher or the small group in mind, while retaining the

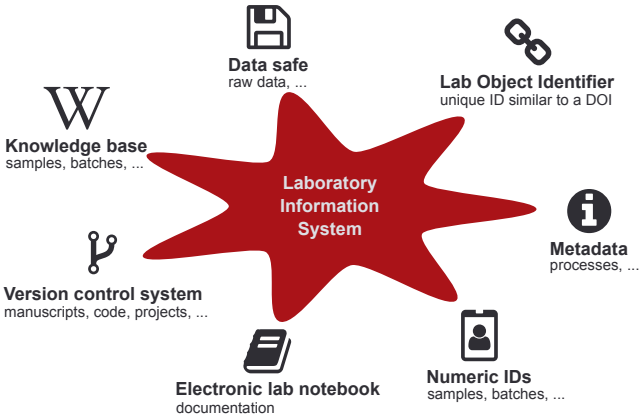

Figure 1. Components of the LabInform modular laboratory information system. A framework for reproducible data analysis is only half the battle if it comes to reproducible research. Additional essential requirements are a way to safely store both, raw data and metadata (data safe) and a method to uniquely refer to a dataset or any piece of information (Lab Object Identifier). Additionally, numeric IDs for samples and alike come in quite handy. A knowledge base and an electronic lab notebook, both based on wiki software and thus easily accessible requiring only a web browser and connection to the intranet, complete the system. Key aspects in setting up the system are its use of well-proven open-source tools combined with a maximum in modularity.

Listing 1. Example of a metadata file containing all information necessary to create a plot spanning several datasets. The file format used here is YAML, mainly because it can be easily read and written by humans.

```
---
format:
  type: ASpecD data analysis
  version: 0.1.0

datasets:
  - loi:xxx
  - loi:yyy

preprocessing:
  - pretrigger_offset_compensation

display:
  - plot_1d

display_parameters:
  xlim: [0 10]
  ylim: [-.05 2]
  xlabel: foo
  ylabel: bar
  aspect_ratio: [4 3]

persistence:
  filename: foobar
  file_format: pdf
```

scalability to larger groups and collaborations between different groups. Besides that, the underlying premise is that information once entered into the system can be converted and fed into other systems and not be lost. To achieve this goal, the system needs to be both, simple and robust, relying on nothing than a small set of best practices and easy to grasp rules in addition to storing information in the most portable and long-lasting formats available.

IV. BASIC CONCEPTS AND COMPONENTS

Before describing the actual components of the system, we would like to state that none of the concepts presented here rely on a particular piece of software. Nevertheless, it seems sensible to name concrete software products that can be and have been used in the given context. If readers take some of the ideas and implement their own systems, the author would be happy to get feedback and hear about the experience. More important, however, is to share the software and ideas with the larger scientific community [22–24].

Fig. 1 provides a first graphical overview of the different components LabInform consists of. Essential concepts contain the data safe as central storage unit for data and metadata ('datasets' in terms of the ASpecD framework [15]), a generalised use of metadata not restricted to information accompanying data acquisition and processing, and unique and stable identifiers for both, samples and alike (numeric IDs) as well as in a much more general fashion all types of 'objects' that can be possibly referred to in a digital context (Lab Object Identifier, LOI). This is complemented by an electronic lab notebook as well as a more general knowledge base for storing information related to the research in a flexible, though structured and easily accessible way. Last but not least, a version control system (VCS) takes care of manuscripts, code, project documents, and alike. Each of these components will be described in more detail hereafter.

A. Metadata

Metadata may occur clearly connected to data, such as in a dataset (as a unit of numerical data and accompanying metadata) or in a lab journal [4]. However, they may as well be used to store arbitrary pieces of information in a structured way. Typical use cases could be the information necessary to create a graphical representation containing data from more than one dataset. Here, a metadata file may contain a list of datasets, a list of preprocessing steps to be performed on the data of each dataset in turn, information on the type of plot to create as well as axes ranges and labels, and eventually details of how and where to store the result. An example of such a file is presented in Listing 1.

Many other possible scenarios immediately come to mind, including a metadata-driven data processing work-

flow as has been implemented independently [25]. The history of each processing step performed on the data of a dataset is stored or can at least be exported to such a metadata file. This ends up with generally two types of metadata files for a given process: One describes what should be done, the other describes what has been done. The only difference between both: the latter includes information on the versions of the software used as well as on the system, time and operator. Depending on the actual context, a metadata file may be stored in the datasafe and thus be accessible using a unique identifier (LOI, see below).

B. Data safe

The data safe is the central storage unit for both, data and metadata (of any kind). Long-term data storage has two aspects: data safety, preventing losses and assuring data integrity, and data security, restricting access to authorised persons. Generally, data in the data safe are stored on the file sytem in a hierarchy of sensibly named directories. To ensure data safety, a network attached storage (NAS) system with redundant hardware is perfectly suited, besides that such system can easily be scaled. Data integrity can be achieved using cryptographic hashes that get stored in files next to the actual data and metadata. Here, at least two different hash values should be stored, one containing only the numeric data, the other spanning both, data and metadata. The reason is that experience shows metadata to change over time, mostly due to inadvertently wrong information that gets corrected upon spotting errors. The numeric data as such should never change. As a by-product of using hashes, datasets can easily be checked for duplicates.

Whereas all metadata should always be stored in text files next to the numeric data, the data safe may use a database for accelerated access and advanced searchability. However, the contents of the database should be automatically created from the information contained in the data safe. The data safe itself will never rely on nor be aware of information stored in a database.

Strategies for handling (temporarily) detached (local) instances of an otherwise central data safe need to be developed, as it will be a regular use case to take the data somewhere away from the lab. Possible solutions range from a read-only instance to mostly automatic merging similar to what version control systems like git provide. Somehow in between these two ranges a solution based on file system synchronisation tools like the well-proven rsync from the Unix world.

C. Numeric IDs

A very simple yet powerful strategy every synthetic chemist, *e.g.*, will be familiar with, is to simply consecutively number certain entities such as samples and batches. Together with a central storage for the information as well as the mapping to the respective number, this allows to easily trace the fate of a sample or batch of substance and alike. In the author's lab, samples and batches are numbered and the information is stored in a wiki forming part of the knowlegde base discussed further below. Eventually, the information is again stored in simple text files with names reflecting the numeric ID. This allows to refer to a sample within the metadata file written during data acquisition simply by its respective number, although still mostly accompanied by a short description. If combined with an electronic lab notebook, this allows to automatically list all measurements performed on a single sample. Furthermore, together with some basic information regarding the respective sample, its storage place and fate can easily be filed.

Another, quite different field from the author's own experience where numeric IDs come in quite handy are DFT calculations. Numbering molecules, geometries, and calculations and connecting each in turn with the others can help to keep the overview of an ever growing amount of computed data.

D. Lab Object Identifier (LOI)

A key aspect of reproducible research is not only to store the full history of processing steps and the software used [15], but to be able to uniquely refer to data and results, fully independent on where exactly they physically reside. Actually, from the point of view of a metadata file similar to the one presented in Listing 1, the actual way the dataset is stored is an unimportant detail, as long as there is a unique identifier allowing to access the information requested. This is similar to the digital object identifiers (DOIs) used to uniquely address electronic documents in the world-wide web (WWW), or more exactly to the Handle system [26] the DOI system itself is based on. This led to the development of a Lab Object Identifier (LOI) as an independent unique identifying system resembling the DOI. Listing 2 shows the general scheme of a LOI together with a more practical example.

An identifier for the publisher is a key aspect of the DOI system that is used for the LOI scheme as well. Everything in a LOI after the publisher ID is fully up to the discretion of the issuer. On the other hand, the system 'knows' how to identify the publisher and can therefore deduce whom to ask for resolving the respective LOI. Such resolvers can be deployed locally on a single computer, in an internal network, and as a globally accessible web service, depending on the respective needs.

A few further comments may help getting more of an idea what could potentially be done with the LOI system. The directory hierarchy of the data safe can be mapped in a one-to-one manner, allowing for a transparent access of datasets. Similarly, entries in the electronic lab notebook as well as numeric IDs for samples and alike can be coded within a LOI. The key aspect of the system is to

Listing 2. General scheme of a lab object identifier (LOI) and a more concrete example. The 42 at the beginning is a rather arbitrary number to distinguish an LOI from other entities of the Handle system, *e.g.*, a DOI. The publisher is coded as four-digit number, followed by a slash. The ID is entirely up to the discretion of the respective publisher. In the concrete example from the author's lab, the LOI refers to a dataset originating from the second TREPR measurement of sample with ID 1.

```
loi:42.<publisher>/<ID>

loi:42.1001/ds/sa/1/trepr/2
```

disconnect the unique identifier from the actual (physical) storage of the corresponding information. Usually, the latter will be stored as file within the file system of a given storage device. Everybody who ever hard-coded a path to a data file in a script or routine for data analysis and afterwards reorganised or simply moved either data or program will immediately see the beauty and power of this approach.

All that is needed is an instance of a resolver holding some sort of a mapping table connecting the LOI with the physical path to the file. Some details of the implementation will be discussed further below.

E. Electronic lab notebook (ELN)

Data are only useful if they are accompanied by additional information, hence metadata. Whereas the metadata accompanying the numeric data should always be stored next to the latter, it is often very useful to have a place to store a 'protocol' of what has been done. This is what the old-fashioned hardcover lab journal that still survived partly today is intended for. The electronic lab notebook (ELN) is an alternative or at least complement offering all the advantages of digital data storage such as easy searchability.

From own experience, an ELN needs to be accessible from every experimental setup used to obtain data. Fortunately, nowadays nearly every setup records data using a computer that is mostly connected at least to the intranet, if not the global internet, and provides a web browser. Hence, using a web-based approach turned out to be very useful. Using some kind of a wiki [27, 28] comes with further advantages. The simple markup language allows for structured and well-formatted entries, besides that every page can be edited just from within a web browser.

Further aspects that can be regarded as 'best practice' from long-term experience are as follows: Create one page for each measurement, even if the sample is measured consecutively, use a web form for creating new pages of the electronic lab notebook, include basic information about the sample, measurement and purpose in a structured way on top of each page, followed by a protocol including the time for each step, and eventually add (automatically generated) graphical representations of the (processed) data. The latter aspect greatly facilitates browsing through the ELN and getting an overview of the outcome of the measurements.

F. Version control system (VCS)

Software for data processing and analysis is often written by scientists themselves and changes over time. While change is perfectly natural for software, reproducibility requires to be able to retrieve the exact version of the software used for a particular processing. Hence, reproducibility requires using a system for keeping track of those changes, *i.e.* a version control system (VCS). Such systems have originally been developed for text-based documentation and are an essential requirement for any kind of professional software development [3, 7, 11, 29]. Different free implementations are available, and distributed VCS such as Git [30] fit best to the academic context. Besides providing a unique link between a distinct version of the software used for an analysis, VCS can be applied in a much broader way, *e.g.* for manuscripts and project documentations.

G. Knowledge base

A last component of the LabInform system closely connected to the electronic lab notebook is a knowledge base containing all information relevant for both, daily work in the lab as well as administrative aspects such as planning publications, grant applications, and commented and well-structured lists of the relevant literature. Furthermore, this is the place for generating and storing numeric IDs for samples and alike as well as the accompanying information. Similarly to the pages for the electronic lab notebook, numeric IDs will be generated using a web form ensuring unique (and consecutive) numbers. Ideally, the knowledge base should be integrated with the electronic lab notebook. At least, the same type of underlying software can and should be used for both.

V. IMPLEMENTATION DETAILS

Whereas the concepts laid out in the previous section are implementation agnostic, the LabInform system is not only an abstract concept, but a reality implemented and actively developed in the author's lab. Hence, a few details of this implementation and the reasons for choosing particular pieces of software will be given below, without loss of generality.

A. Metadata

Metadata need to be stored in a text format that can be simply read by a computer, while retaining human readability as well. For all metadata that shall be easily readable by users, YAML [31] turns out to be the best choice, as it comes with the least amount of formatting. Exchanging data between programs may well be done using other formats as well, be it JSON or even XML. Automatic conversion between these files is straightforward and implemented in many programming languages including Python. For a thorough discussion of file formats, the reader is referred to [18].

B. Data safe

Generally, the data safe is not much more than a hierarchy of directories containing all datasets and stored on a NAS that may additionally be frequently backed up. Creating and storing cryptographic hash values for each dataset can be and has been automated using bash scripts, assuming a unixoid operating system being used. User access control can be implemented on the operating system level, but will eventually be best solved using a web interface and an underlying web service for accessing the data safe using either a web browser or programmatically from within routines for data processing.

C. Lab Object Identifier System

The scheme for the actual IDs is entirely up to the discretion of the respective publisher. Hence, no further details will be presented here. For the time being, there will be no organisation issuing and managing publisher identifiers. Still, the system is intended to be used within the setting of a group, or perhaps even within a network of collaborators, but not on a global scale. Needless to say that in case such global system once emerges, mapping (local) LOIs to the newly created unique IDs should be straightforward.

The entity responsible for mapping a LOI to the actual path to the file containing the information, hence the resolver, can be set up as a very simple system to begin with. The most primitive implementation regarding access to the data stored in the data safe using LOIs would be to simply reflect the data safe directory hierarchy in the LOI. Thus, any externally maintained mapping table becomes fully obsolete, rendering the system rather robust. Single files as a mapping table may easily get lost. An entire directory structure, however, crafted from well-chosen names, is much more unlikely to accidentially vanish. In a broader and more complex environment, some kind of automatically generated lookup table and a Docker container running the web server nginx serving both, a very simple web form for entering the LOI as well as a web service accessible from within programs for data analysis and processing [15] are fully sufficient.

D. Electronic lab notebook and knowledge base

Important criteria for choosing a wiki software for both, electronic lab notebook and knowledge base, are portability as well as a small footprint, together with free availability as open-source software under a liberal license and being built using well-established programming languages having a large and vivid user basis. Dokuwiki [32] turned out to be an excellent choice in this respect. Pages are simple text files stored in the file system, no database is necessary, and a dokuwiki instance can even be run from a memory stick. Even without a running dokuwiki process, the information can easily be obtained provided access to the underlying file system can be ensured.

From the wealth of plugins available, a combination of bureaucracy and structured data plugin has been proven very useful for creating entries and pages both, for the electronic lab notebook and entities with numeric IDs such as samples and alike, using simple web forms. Here, using the structured data plugin is preferred over its more powerful successor, as the latter stores some information exclusively in a data base, jeopardising both, portability as well as long-term access. Several plugins for handling literature databases stored in BibTEX files are available as well greatly facilitating the maintenance of commented and structured lists of relevant literature.

E. Version control system (VCS)

A distributed VCS fits best to the context of academic research in the individual laboratory scale. We strongly recommend using Git [30] due to its widespread use and well-developed tool chain and integration in different other programs and workflows. As a graphical, web-based frontend, Gitea [33] comes in quite handy due to a very small footprint. Gitea as a frontent can be deployed readily using official Docker images. The reason for installing a local VCS instance rather than relying on cloud-based solutions such as GitHub and GitLab is twofold: independence of external infrastructure and (even more important) control over the own data and their access.

F. Containerisation using Docker

Modularising all the components described so far in a platform-independent way remains the last aspect to be mentioned. All the different components can be put into Docker containers [34]. This greatly enhances modularity and platform independence while minimising the administrative effort necessary during routine operations.

Whereas this approach has been adopted in the professional software development particulary for web services, it has been proposed for computational reproducibility as well [35]. Besides the explicit components of LabInform, this extends as well to the programs used for data analysis and processing, *e.g.*, systems based on the ASpecD framework [15]. Modular and step-wise introduction of the LabInform system to an existing infrastructure and workflow thus become possible, as do scalability and further development parallel to productive use without impairing each other.

VI. CONCLUSIONS

Taken together, we have outlined general strategies for a modular laboratory information system built from open source components that can be implemented in a typical scientific setting from a single researcher to a larger group 'on the go'. Furthermore, we have presented concrete solutions for each of these components that are readily available. In summary, the system presented here, LabInform, helps researchers to enhance the reproducibility of their research with a minimum of necessary additional effort regarding hard- and software as well as manpower, and ensuring scalability, robustness and sustainability.

ACKNOWLEDGEMENTS

The ideas presented have evolved over more than a decade, and many people have helped shape the ideas and implemented parts of programs eventually resulting in the larger infrastructure described here. To name just the most important persons in chronological order: B. Paulus, D. Meyer, J. Popp, M. Schröder. Thanks to all colleagues and friends who readily discussed these aspects and for their patience with the author, and R. Kaal in particular. A lecture that grew out of the author's preoccupation with the ideas presented here turned out to be an ideal test bed. Thanks to all students actively attending it. The German Research Foundation (DFG, Grant BI-1249/3-1) is gratefully acknowledged for financial support.

SOFTWARE AVAILABILITY

The third-party components the LabInform framework is largely built upon are all available open-source and free of charge. Those components specifically developed are equally available open-source and free of charge under a BSD license. Details will be published on the respective website https://www.labinform.de/ together with a detailed documentation and links to the other resources required. A demo instance of both, ELN and knowledge base can be found under https://demo.wiki.labinform.de/. Additionally, Docker containers including each of the components will be provided.

[1] Chalmers, A. F. *What is this thing called Science?*, third edition ed.; Open University Press: Berkshire, UK, 1999.
[2] Mesirov, J. P. Accessible reproducible research. *Science* **2010**, *327*, 415–416.
[3] Sandve, G. K.; Nekrutenko, A.; Taylor, J.; Hovig, E. Ten simple rules for reproducible computational research. *PLoS Comput. Biol.* **2013**, *9*, e1003285.
[4] Goodman, A.; Pepe, A.; Blocker, A. W.; Borgman, C. L.; Cranmer, K.; Crosas, M.; Di Stefano, R.; Gil, Y.; Groth, P.; Hedstrom, M.; Hogg, D. W.; Kashyap, V.; Mahabal, A.; Siemiginowska, A.; Slavkovic, A. Ten simple rules for the care and feeding of scientific data. *PLoS Comput. Biol.* **2014**, *10*, e1003542.
[5] Michener, W. Ten simple rules for creating a good data management plan. *PLoS Comput. Biol.* **2015**, *11*, e1004525.
[6] Peng, R. D. Reproducible research in computational science. *Science* **2011**, *334*, 1226–1227.
[7] Osborne, J. M. et al. Ten simple rules for effective computational research. *PLoS Comput. Biol.* **2014**, *10*, e1003506.
[8] Merali, Z. ...why scientific programming does not compute. *Nature* **2010**, *467*, 775–777.
[9] Goble, C. Better software, better research. *IEEE Internet Comput.* **2014**, *18*, 4–8.
[10] De Roure, D.; Goble, C. Software design for empowering scientists. *IEEE Softw.* **2009**, *26*, 88–95.
[11] Wilson, G.; Aruliah, D. A.; Brown, C. T.; Hong, N. P. C.; Davis, M.; Guy, R. T.; Haddock, S. H. D.; Huff, K. D.; Mitchell, I. M.; Plumbley, M. D.; Waugh, B.; White, E. P.; Wilson, P. Best practices for scientific computing. *PLoS Biol.* **2014**, *12*, e1001745.
[12] Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28*, 235–242.
[13] Berman, H.; Henrick, K.; Nakamura, H. Announcing the worldwide Protein Data Bank. *Nat. Struct. Biol.* **2003**, *10*, 980.
[14] Bernstein, F. C.; Koetzle, T. F.; Williams, G. J.; Meyer, E. F., Jr.; D.Brice, M.; Rodgers, J. R.; Kennard, O.; Shimanouchi, T.; Tasumi, M. The Protein Data Bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* **1977**, *112*, 535–542.
[15] Biskup, T.; Popp, J. ASpecD: A modular framework for the analysis of spectroscopic data focussing on reproducibility and good scientific practice. to be submitted.

[16] Whitehead, A. N. *An Introduction to Mathematics*; Dover Publications: Mineola, 2017.

[17] Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* **1991**, *23*, 5–48.

[18] Raymond, E. S. *The Art of UNIX Programming*; Addison Wesley: Boston, 2004.

[19] Martin, R. C. *Clean Architecture. A Craftman's Guide to Software Structure and Design*; Prentice Hall: Boston, 2018.

[20] Hunt, A.; Thomas, D. *The Pragmatic Programmer*; Addison-Wesley: Boston, 1999.

[21] Schwab, M.; Karrenbach, M.; Claerbout, J. Making scientific computations reproducible. *Comput. Sci. Eng.* **2000**, *2*, 61–67.

[22] Barnes, N. Publish your computer code: it is good enough. *Nature* **2010**, *467*, 753.

[23] Ince, D. C.; Hatton, L.; Graham-Cumming, J. The case for open computer programs. *Nature* **2012**, *482*, 485–488.

[24] Prlić, A.; Procter, J. B. Ten simple rules for the open development of scientific software. *PLoS Comput. Biol.* **2012**, *8*, e1002802.

[25] Freire, J.; Silva, C. T. Making computations and publications reproducible with VisTrails. *Comput. Sci. Eng.* **2012**, *14*, 18–25.

[26] Sun, S.; Lannom, L.; Boesch, B. *Handle System Overview*; RFC 3650, 2003.

[27] Leuf, B.; Cunningham, W. *The Wiki Way. Quick Collaboration on the Web*; Addison-Wesley: Upper Saddle River, NJ, 2001.

[28] Mader, S. *Wikipatterns*; Wiley Publishing, Inc.: Indianapolis, IN, 2008.

[29] Taschuk, M.; Wilson, G. Ten simple rules for making research software more robust. *PLoS Comput. Biol.* **2017**, *13*, e1005412.

[30] Chacon, S.; Straub, B. *Pro Git*, 2nd ed.; Apress: New York, NY, 2014.

[31] `https://yaml.org/`.

[32] `https://www.dokuwiki.org/`.

[33] `https://gitea.io/`.

[34] `https://www.docker.com/`.

[35] Piccolo, S. R.; Frampton, M. B. Tools and techniques for computational reproducibility. *GigaScience* **2016**, *5*, 30.