Open Science and Neuroimaging

A Practical Guide

Index

If you have any ideas about how this guide could be improved, feel free to suggest it in this Google Doc.

1 Introduction

Open science has seen a steep rise in the recent years. However, when I first started thinking about how to do reproducible research, I quickly became overwhelmed by the things you can and should do. By writing this guide, I wanted to bring order into this mess and provide an overview of the tools and practices and how users can implement these open-science routines into the everyday research workflow. In the world of neuroimaging, and especially functional Magnetic Resonance Imaging (fMRI), a lot of attempts have been made to simplify reproducible research.

The goal of this guide is to provide an introduction to how to conduct neuroimaging studies in an open science framework. The guide will be structured by the four principles of **Open Source**, **Open Methodology**, **Open Data**, and **Open Access**. For each part there will be a description of the available tools (software, platforms, etc.). You will find links to tutorials, downloads and in-depth information along the way. In the end there will be a short conclusion with a few tips on where to begin your journey into the world of open science.

1.1 What is Open Science?

Open Science is the idea that science should be as transparent and comprehensible as possible, making it accessible to everyone. It is based on the principles of Open Methodology, Open Source, Open Data, and Open Access (Kraker et al., 2011). Sometimes the principles open peer review and open educational resources are also mentioned (Kasberger, 2013). The first four principles will be the structure for this guide. But before we delve into the practical aspects, let us have a look at each of them first.

Open Source. Brain imaging has become a data heavy discipline. A lot of different software exists that can help researchers make sense of their data which differs in terms of licensing. On the one side of the spectrum there is software that is part of the public domain, and by that anyone is allowed to use it freely, modify it and redistribute the software and code. Then there are permissive licenses differing in how restrictive they are regarding usage, modification and redistribution. These are the licenses often used by open source software. On the other side of the spectrum there is proprietary software, which usually costs money and where all rights are reserved (“5 types of software licenses

you need to understand | Synopsys", 2016). Examples of this would be Microsoft Word being proprietary, in that you have to pay for it and are not allowed to redistribute it in any way, and Libre Office, which is open source and completely free.

Software not only differs in terms of licensing, but also in terms of documentation. As mentioned, open science strives to be as transparent as possible. In fMRI and in brain imaging in general a lot of complicated procedures are applied to the data in order to end up with interpretable results. This makes it even more important for researchers to understand what the used software is actually doing. This is why a thorough documentation is another important aspect of the principle of Open Source. For the tools mentioned in this guide this can usually be found online and will be linked to.

Open Methodology. Open Methodology can be summed up as the effort to provide all the information necessary to reproduce a study. Although articles in scientific journals always contain a methods section, it rarely contains all the information that one would need to reproduce the study step by step (Kraker et al., 2011). One way to make this possible is to practice open notebook science. This basically means that everything that was formerly documented in the lab notebook (e.g., experimental procedures, measuring parameters, code etc.) will be made public, usually in an electronic form ("Open-notebook science", 2019). In a discipline where there is a lot of data processing, it is important to share the analysis scripts, so that the data analysis can be made even more transparent.

Another aspect that falls into the area of Open Methodology is that of preregistration. Preregistration means that a study is registered at a journal *before* it is conducted. The journal guarantees publication of the study, regardless of significant effects being found. . This approach is there to prevent publication bias (also termed the "file drawer problem"; Rosenthal, 1979), bad research practices such as p-hacking and cognitive fallacies that a researcher might fall into (Nuzzo, 2015).

Open Data. Research is only maximally transparent if all data are shared as well. As in neuroimaging there are large amounts of complex data, the question for the last years has been in what way imaging they should be shared. Today there are a number of platforms that a researcher can choose from to upload their data. Some of them are more curated than others, allowing only data in a specific format to be uploaded.

Uploading data from participants also brings some legal issues, which will also be addressed in the chapter on Open Data. These questions revolve around the anonymization of imaging data and what else is necessary in order to publish them.

Open Access. Journals with Open Access enable everyone to profit from the results of research. There are different kinds of Open Access journals, namely green Open Access (journal allows the publication of preprints) and gold Open Access (the peer-reviewed version of the article is accessible by everyone). This chapter will discuss the pros and cons of Open Access publication and different alternative models of publication.

1.2 Why Open Science?

Open Science has several goals it wants to accomplish. These can be divided into value-based goals, goals that have to do with the process of conducting science, and output-related goals.

On the value side, there are things like inclusiveness and Open Access. Science should be accessible for everyone. With the subscription-based model of scientific journals, this is not always the case. Especially in developing countries, some universities might not have the monetary means to subscribe to every journal, which could lead to some researchers not being able to read up on all the research that is relevant for them.

This also connects to the principle of open educational resources, which can be seen as another value-based goal. Making those resources available online enables people to educate themselves on a level that would otherwise not have been affordable for them. This is a practice that Ivy-League universities like Harvard, the Massachusetts Institute of Technology, or Oxford have been practicing for a while now. A lot of their course materials, especially in the natural sciences, math, and computer science are openly accessible for everyone (see for example edx or coursera).

An example for process-based goals is to make science more transparent and therefore results more easily reproducible. Especially in psychology, a lot of results could not be replicated (Open Science Collaboration, 2015). This has been explained by bad research practices and the need for researchers to publish positive results in order to advance their career (Begley & Ioannidis, 2015). In order to solve this problem, researchers should make

every step of their research transparent. This would not only help other researchers to replicate a given study, but also improve the quality of research, as bad practices can not be hidden anymore.

An output-related goal is to make the results of research openly accessible. This does not only include the results that are reported in an article, but also the raw data that these results rely on. These data might also prove to be valuable for other research questions. By reusing data, redundancy is reduced and the available resources can be used more efficiently.

The goals listed so far aim at why Open Science might be valuable for science. But is it also valuable for you as a researcher? McKiernan et al. (2016) argue that it is indeed. First of all, publishing Open Access will lead to more citations. This might in part be due to the fact that Open Access publications also get more media coverage, and that they can be shared on social media (McKiernan et al., 2016).

Apart from your research getting more attention, it will likely also have a higher quality. Documenting everything in a way that others can understand forces you to be less sloppy. This can benefit you in the future: If you have to look back at your analysis at a later point in time, a good documentation will vastly improve your ability to retrace your steps. Simply put, this means less work in the long run.

Sharing scripts and data also makes collaboration easier. When researchers from other universities can easily access your work, the psychological barrier to reach out to you is lower than if you do not share anything.

To sum it up, there are a lot of reasons for Open Science, both for the sake of doing good science and how it can benefit your career as a researcher. Now that we have covered the '*why'*, we can next look at the '*how'*, beginning with the principle of Open Source.

2 Open Source

An important part of Open Science is the use of software that has well documented and openly accessible source code a. This does not only mean using software that is Open Source in itself, but also that analysis scripts are to be shared. Additionally, the software should preferably be free of charge and run on any computer (platform independent software).

This sounds like a lot, but thanks to the effort of various researchers it is possible to do your data analysis with software that adheres to all the aforementioned principles. While the main part of this chapter will deal with the question of how to do data analysis in a reproducible manner, I will also explain a little bit about PsychoPy, an Open Source software that can be used to generate stimuli for psychological experiments, making it suitable for imaging studies as well.

2.1 Analysis of Imaging Data

For our analyses to be open, they need to run on any system while always giving the same results. This can be achieved using a technology called virtualization. Virtualization means that a piece of software, e.g. an operating system, is run as a virtual computer *inside another operating system*. This way, you can for example run Linux on a computer that normally runs on Windows. This would usually be done using a software like VirtualBox that sets up a virtual machine. The virtual machine then acts like a real computer and it can even interact with the operating system it is embedded in.

For our purposes we make use of containerization, which is a special case of virtualization. Containers are in essence small virtual machines. Though there are some technical differences, the important difference for us is a philosophical one: instead of setting up an operating system that contains software needed for all kinds of purposes, a container only contains the software necessary for a specific task. With the help of Docker (see section 2.1.1), a self-contained system can be set up that has all the software that we need to run our analyses. This also enables us to make use of functions from several analysis software packages using Nipype, independent of our operating system. Nipype works as an interface between these packages and is based on the open-source programming

language Python. We will write the scripts for our analysis in Jupyter Lab & Jupyter Notebooks, which offer a good way to have code, documentation and output all in one place. Additionally we will encounter the brain imaging data structure, short BIDS. This is an approach to standardize the arrangement of imaging data. Because it is standardized, programs called BIDS-apps can make use of this, offering additional functionality.

I will describe these tools in more detail in each of the subsections, so you will gain a clear understanding what everything does, and how it fits in the analysis workflow.

2.1.1 Docker

Docker is a containterization software. As mentioned above, containerization works similar to a virtual environment, in the sense that you simulate an operating system that is different from the one you are using on your computer. It runs in a container. A container is a self-contained system that has everything that is needed to run software: An operating system, a file system and code. Docker enables us to construct such a container and run our analysis in it. Singularity is another containerization software that is often used to run containers in a shared IT infrastructure or a cloud. This might come in handy for running analyses that require much computing power.

While in theory both softwares can be run in the cloud, often Singularity is preferred for this purpose due to security issues. With Docker, the user has full access to the host system, which can lead to unwanted outcomes such as accidentally deleting or overwriting files. In this guide I will concentrate on Docker, as it would go beyond the scope of this guide to give an introduction to both. Singularity is however quite similar to work with, so that it shouldn't take too much effort getting to know it after learning Docker.

Why should we run our analysis in a Docker container? Using containerization software has several advantages. First, we can not only share the scripts of our analysis, but also the complete computing environment, meaning all software (in the specific version) and dependencies to run the analysis. Second, containers run on any operating system. Collaborating on projects is possible even when one researcher uses Windows, the second Mac OS and the third Linux. Third, the analysis will always deliver the same results. There are no differences due to different software versions or differences in the

underlying Kernel (the core of a computer's operating system). These advantages make containerization such an important technology for achieving reproducibility.

With Docker, we can set up a data processing environment with the needed software for data processing in something called a Docker image. The image contains all the software that we need for our analysis. The container is the instantiation of our Docker image. Docker images can then be shared on the internet on a platform called Docker Hub. Other researchers can download these, for instance to use it on their data. Downloading an image is called "pulling".

If you want to know how to install Docker on your system, please see this short tutorial here. It is part of a very good tutorial on Nipype. You will read more about Nipype in section 2.1.5.

2.1.2 Neurodocker

In order to make a Docker image, we need to make a Dockerfile, which is basically a recipe for how to build our container. Fortunately, with Neurodocker there is a tool that makes this task very simple. It runs directly in Docker as a Docker image. See the part on Neurodocker in the Nipype tutorial on how to use it. There is also a list of examples on the Neurodocker Github page.

2.1.3 Python

Python is an open-source programming language. The current version is Python 3. It is quite beginner-friendly, but also very versatile and powerful, making it suitable for many scientific appliances. As it contains packages for mathematics, scientific computing, statistics and plotting, it is a good all-around choice to use for data analysis. It also allows for more sophisticated methods such as neural networks and machine learning. Whatever your goal may be, be it to just be able to set up a basic reproducible analysis pipeline, or if you are willing to dive deep into the world of programming, Python is a good place to start.

But how to learn Python? There are a lot of options, and it depends on your goals and what works best for you what you should opt for.

Here are some options on Python for science:

A tutorial for Python in scientific contexts are the Scipy lecture notes. The best thing about it: There are a lot of exercises. Exercises are great—they are the best way to actually learn programming! Also it gets updated regularly, so that what you learn is up to date.

- Go for this option if you want to dive deep into the world of scientific computing with Python.

The Nipype tutorial contains a short introduction to Python that covers everything to use Nipype (see the Nipype chapter on how to run it interactively).

- Go for this option if you just want to learn the basics in order to be able to use Nipype.

Some additions:

Tutorial on data visualization in Python. Gives some additional options on how to make graphs.

- If you have no experience in programming whatsoever, you might want to consider the book Learn Python 3 the Hard Way. The author follows the philosophy that in order to learn a programming language, you have to type everything yourself. It contains a lot of exercises and video material. Working through this book will give you a strong foundation to build further programming skills on. Downside: You will have to buy the book.

2.1.4 Jupyter Lab & Jupyter Notebooks

As we have seen, Python is a programming language. While it is possible to write your analysis scripts in plain Python, there are actually nicer ways of doing this, so that the scripts are easily readable and understandable by others. One way to achieve this is by using Jupyter Notebooks. In a Jupyter Notebook you can write and run code, directly see the output, and write documentation in the easily readable markdown language. It runs in your browser (Firefox, Chrome or Safari) and is available for macOS, Windows and Unix-based systems.

The term Jupyter Notebook refers both to the file type, also termed Notebook files or notebooks, and the computational environment. The second however is now outdated and has been replaced by Jupyter Lab (for an overview of the differences between Jupyter Lab

and Jupyter Notebook, see this link). For our purposes, the notebook will usually run on Python, though it is possible to use another Kernel, for example MATLAB, or R.

But why should we use it for data analysis? The main aspect is to simplify the documentation process. Instead of writing comments in your code, we can write formatted text in markdown, which offers functionality for formulas, emphasis, links etc. (for a quick overview on how to format text in markdown, see for example here, or just google 'markdown cheat sheet'). Second, we can directly see the output our code generates, which makes working with it very intuitive. Third, you can run Jupyter Lab in a Docker container. This enables us to share not only well documented code, but also directly provide the computational environment needed to run it.

2.1.5 Nipype

So far, this chapter has been dealing with data analysis in general. But how should we apply it to neuroimaging data? Here Nipype comes into play. Nipype is a Python-based interface for building neuroimaging analysis pipelines that allows the combination of different analysis software packages (including, but not limited to ANTS, SPM, FSL, FreeSurfer, Camino, MRtrix, MNE, AFNI, Slicer, DIPY). It offers an easy and standardized interface for all those packages. That way, if you know how to use Nipype, you can mix and match from all of them.

I will not go much into detail about how to work with it, as there are excellent and extensive tutorials on Nipype which you can find here. The tutorials can either be run locally in a docker container or online in something called a binder. For the binder, click on the link tagged "binder service" in the introduction text of the linked website. Should you want to do the more advanced tutorials, I would advise to use the docker container instead of the binder, as the latter only runs on restricted resources. Follow the instruction given here to run the tutorial in a Docker container. The tutorial also contains short introductory bits on related concepts such as Python, BIDS, and Jupyter-Notebooks.

2.1.6 BIDS

BIDS stands for brain imaging data structure. Since its introduction in 2015 it has become a standard on how to arrange imaging data in folders and subfolders, including naming

conventions. Before BIDS, there was no consensus on how to structure imaging files. One can imagine that this lead to a lot of confusion. Furthermore, scripts that worked for one data set would not work for another, as the files were arranged differently.

Having a standard solves this problem: Analysis scripts that are based on BIDS are now more or less universally applicable. This is why using the BIDS-format offers a number of advantages ("brain imaging data structure", n.d.):

- Collaboration is easier. Different researchers can work on the same data set without any need for explanation on how it is arranged.
- You can use BIDS-Apps to analyze your data. BIDS-Apps are data processing pipelines that understand the BIDS format (more on that below).
- Data sharing is easier. Neuroimaging databases such as OpenNeuro.org require the data to be uploaded in the BIDS-format, so that they are understandable for everyone. For more regarding data sharing see the chapter on Open Data.

But how do we make our data BIDS-conform? Luckily, there are a number of converter tools that will do that conversion automatically (e.g. HeuDiConv for fMRI, plus a tutorial here). In the context of fMRI, you will generally need the original DICOM files for these to be able to work. Find the full list of converters from various formats on the BIDS website.

Now that you have converted your data, you might want to check if the conversion actually worked. For this, you can use a BIDS-App called BIDS-Validator. Additionally to checking the quality of the conversion, it tells you if there are any missing values in your data set. It can be run directly in a browser (Chrome or Firefox): To do this, visit this website and select the folder to your data set. Don't worry—no data is uploaded. Alternatively you can run the BIDS-Validator in the command line, in Docker or in Python (see here how to do that).

But BIDS-Apps can do a lot more than that, for example:

- Quality Control: MRIQC. This is short for Magnetic Resonance Imaging Quality Control. See here how to install it. MRIQC will compute a bunch of Image Quality Metrics both for the structural and functional images. They are summarized in interactive graphs, making it easy to find outliers in the data. You can try out how these visual reports work here (click one of the two links under "Visual Reports").

- Anonymization: BIDSonym. De-identifies fMRI images using one of several algorithms. They all basically crop the part of the image that contains the face, so that it can not be reconstructed and used to identify the person. De-identification is needed when you want to share your imaging data.
- Preprocessing: fmriprep. Yes, BIDS-Apps can even do the whole preprocessing for you. The philosophy of fmriprep is that of a glass box: the processing steps are automated and standardized, but you can see exactly what is going on because of quality reports and a thorough documentation. It uses the best tools that are currently available, by combining algorithms from different software packages. See also the tutorial on how to run it in Docker, and the documentation.

2.2 Stimulus Generation

A lot of this chapter dealt with data analysis. But before there is data to be analyzed, it must first be collected. In the case of fMRI, this happens in the form of experiments that usually show stimuli with a specific timing. Specialized software is needed for that, and the Open Source alternative is PsychoPy.

PsychoPy is based on Python and consists of two main parts: the Builder interface and the Coder interface. The Builder is a graphical user interface (GUI) where you can construct your experiment visually in a modular manner. It lets you define trials, build loops, and import stimuli. You can find a video that explains the basics of the builder here. It is also possible to compile your experiment into a Python file at any time. This lets you examine the code that your experiment runs on in the Coder interface. However, if you make changes to this file, this will not affect the experiment that you made in the Builder. You can choose to continue to work in the coder, but you can not go back to working on this file with the Builder. Basically, you can easily switch from Builder to Coder interface, but not the other way around.

Compiling your experiment as code is a good way to get started working with the Coder. The GUI of the Builder interface will help to get an intuition of how experiments are constructed. But chances are that at some point you will want to use some functionality that is not possible to implement using the Builder alone. That is why I would advise to get comfortable with scripting early on. On the PsychoPy website there are some tutorials to

do so. Start with the basic concepts to get an understanding of the principles that PsychoPy scripts are based on, then proceed to the PsychoPy tutorials to see them applied.

3 Open Methodology

As you already know, one of the goals of the open science movement is to make science reproducible. In 2015, the Open Science Collaboration (2015) tried to replicate 100 studies. Only about half of these showed the effect that was originally found. This has been explained both by the need to publish in order to stay relevant as a researcher, and bad research practice, which might be a result from the publish or perish attitude in the first place (Begley & Ioannidis, 2015).

In order to ensure high quality science, it is therefore necessary to make everything accessible that is needed to replicate a study. This is also advised by the Deutsche Forschungsgesellschaft (2019), who states in their codex for good scientific practice that researchers should publish data, methods, and software, as well as code and analysis scripts to make it accessible to the public.

In order to be able to replicate a study, one would probably start by looking at the methods section. However, from my own experience I can say that this does not always suffice. Decisions about study design are not always explained and in my case, the code necessary for data analysis was not made public. This lead to a lot of guessing and trial and error.

We have established that there is a need for sharing detailed information on the methodology that was used in a study. In particular, it is important to share code and analysis scripts in an accessible way. How can we do that, so that others may benefit from it the most? This is the question I will try to answer in this chapter.

3.1 Open Lab Notebook

In the natural sciences, especially where experimentation takes place in a lab, every step of the experiment is noted in a lab notebook. Traditionally, this was a book that lay around in the lab, where entries were made by hand and pictures were glued in. With the invention of electronic lab notebooks, it is possible to do this digitally. This has the advantage that data and graphs can be directly displayed and sharing it has become more easy.

The open lab notebook goes a step further. Practicing open notebook science in its most extreme form means sharing this lab notebook “live”, documenting every step while doing

a study. By doing so, research becomes maximally transparent, thus enabling other researchers not only to replicate a study, but to retrace the whole process from conceptualization to publishing.

This of course also has some disadvantages. One danger this poses is that other researchers might steal an idea and publish it before you are able to do so. This practice is known as scooping, though it is not known how often it actually happens. Scooping can be circumvented by placing an embargo on the lab notebook, making it public only after the paper is accepted at a journal.

Another issue might be that some journals might count a lab notebook as a publication, meaning that they will not accept the paper for their journal. However, as most journals also accept publishing preprints—see for example here for a short list of journals and their preprint policies—this might not be as big of a problem as it seems at first. In any case, if you should opt for an embargo period of your lab notebook, you can check the requirements of the journal before you decide to make it public (see the chapter Preprints to find out how to do that).

When writing an open lab notebook you should also be cautious about sensible data of participants. Every information should be anonymized or pseudonymized before making it public. What is on the internet, stays on the internet, so always check twice before posting anything online.

Open notebook science can be done on different levels:

- *Planning and conceptualization*. Writing down your general progress at least on a weekly basis is a good practice. You might not know a month in advance why you made a specific decision in your study design, so being able to go back and have a look at your thoughts from back then can really come in handy. You might not be comfortable to share this publicly, as ideas take time to form and making errors is part of the process—and that is alright. Practicing open science is not “all-or-nothing”, but a gradual process. You might decide to share this just with the people you work with on your current project. Whether you make it public or not, keeping a “research diary” might really help you in your research progress.

- *Experimental design*. This is more similar to the classical lab notebook, where you write down experimental procedures and the like. In a paper, this would be what you would write in the methods sections. As space in journal articles is often very limited, sharing this information in a more detailed way can help others to replicating your study.
- *Analysis scripts.* This is probably what others will benefit from the most. Sharing your analysis scripts that are coherent and well-documented does not only help others but also yourself, when you are trying to retrace what you did in your analysis at a later point in time. The option to insert formatted text in your Jupyter notebooks is a good tool to make a well structured and comprehensible documentation of your analysis with more thorough explanations than what would be possible in the methods section in a scientific publication. You can share these on Github, a platform that allows you to upload code and work on it collaboratively.

3.2 Open Science Framework

When researching open science, sooner or later you will stumble upon a website called Open Science Framework (OSF). OSF is a platform that ties together a bunch of open science functionalities. Researchers can create projects with different components, such as an open lab notebook, data, code, and so on. You can do a preregistration of your study here (more on that in the chapter Preregistration), upload preprints, and upload your data (with unlimited space!).

You can use all these functions directly as part of the website. However, what makes OSF so useful is that it offers integration to a large number of other platforms. For example, if you use Github to upload your code, you can directly tie it into your project. If you upload a preprint on bioRxiv, it also gets listed on OSF, and you have the option to connect it to the project it belongs to. OSF also offers integration of referencing software such as Zotero, so that it automatically synchronizes your bibliography as the project progresses. You can also create a Wiki for your project. If you want to upload a poster you can use Figshare to upload the pdfs and integrate them too.

Even if you are not interested in all the functions OSF has to offer, it might still be worth it to list your project on the platform. Due to its integration of various other tools, it can serve

as a connection point to your data, source code and so on, making your research more discoverable.

3.3 Github

Github is a platform where you can upload your code and share it with others online. It makes use of the version control system Git, meaning that it keeps track of changes made to the code. More specifically, every time you "commit" changes to your code, it records what changes were made, when, and why—"why" being a description of the changes you write yourself every time you make a commit.

Github is not just a version control system, but a *distributed* version control system. This means that it keeps track of another w-question: Who. Several people can work independently on the same project and make commits to it. Projects are organized in repositories, which basically are collections of code. People can "pull" the repositories to their computers (which is Github-lingo for downloading them). They then make changes, add files etc., in order to then "push" them to the repository.

Sometimes, people will make changes to the same files. As there are two different versions of the file now, the changes will have to be "merged". Oftentimes, Github will do this automatically. However, there will occasionally be a "merge-conflict". Then someone will have to look at the code and decide what changes should be kept in the version that will be stored on Github.

You will find some tutorials from Github here. These will show you how to use Github using the command line. A bit more intuitive for the beginner is using a desktop client such as GitKraken.

GitKraken is a desktop client for Github, which is available for all platforms. It shows the project tree visually, which will help to get an intuition of the versioning system of Github. You can download it here, and do a short tutorial here.

Sometimes, you will just want to download some code from someone else which they have stored on Github. While there is an option to download the whole repository as a zip-file, Github does not directly support downloading only subfolders. You will have to use a little workaround to do this: Copy the link to the subfolder your want to download, go to

DownGit and paste it. Then click download, and you will be able to download the folder as a zipped file.

You will need an editor to make changes to these files. There are many open available editors. One example is Atom, which is platform-independent and also has a Git module installed, which means you can push your changes directly from Atom, among other things.

3.4 Referencing

When doing research, you have to keep track of a lot of literature. Using a reference manager makes this job a lot easier. Though a bunch of proprietary software is still used in the academic context, there are viable Open Source options available. When looking at the comparison of reference management software on Wikipedia, one software seems to dominate both in terms of being free and offering a whole lot of functionality: Zotero.

I have been using it for writing this guide, and especially the web client is really helpful. With just one click it collects all information necessary for citation and stores it in your library. Additionally, it automatically downloads the PDF, if available, or makes a snapshot of the website you want to store. You can easily collaborate, as there is no limit on how many people can work together on one project. You also have the option to synchronize your library with Zotero's online storage. You can even sync all the files with it, the only limitation being that free storage space is limited to 300mb. This limitation does not exist for the references themselves, only for the PDF files attached to them.

Another advantage of using Zotero is that you can link your library to a project on OSF. Added references will automatically be synchronized to your project. To set this up, go to your project on OSF and click on the Add-ons tab. Scroll down to Zotero in the list of available add-ons and click enable, then follow the instructions given. You can download Zotero here.

3.5 Preregistration

An important aspect of Open Methodology is that of preregistration. Preregistering your study means that you make your hypothesis public before conducting the study. You announce the dependent and independent variables, the criteria for each condition that

your experiment will consist of, and, importantly, you list the analyses you plan to conduct. This way there can be a clear distinction between confirmatory and exploratory data analysis, preventing bad research practices such as p-hacking.

Doing a preregistration of your study offers several advantages. Firstly, it forces you to think deeply about your study design before beginning with data acquisition, as the process of preregistration requires you to formally write everything down. This will likely improve the quality of the study.

Secondly, as mentioned, it is a way to prevent bad research practices, as no analyses that were actually exploratory and not planned beforehand can be disguised as confirmatory.

Thirdly, it takes some of that vulnerability that open research entails. In the section on the practice of doing an Open Lab Notebook, I brought up the possibility of scooping. Preregistering a study makes scooping unlikely, as it gives you proof that you were the first to have a research idea. Preregistrations have a digital object identifier (DOI), by which they can be cited, and once published, they can not be changed.

If you want to do a preregistration of your study, you have several options. The easiest will probably be AsPredicted. You will only need to answer nine questions, as it asks “only what's necessary to separate exploratory from confirmatory analyses” (“AsPredicted: Home”, n.d.). See also here for what to take note of when doing a preregistration. You can link your preregistration on AsPredicted to an existing OSF project.

Speaking of OSF: The platform also offers a number of ways to do preregistrations. Apart from their own template, which is a lot more detailed than the one from AsPredicted, there is a large number of other templates you can choose from. As the choice can be a bit overwhelming, I would advise to stick to the two options I mentioned: AsPredicted if you want to register only what is necessary, OSF if you want to do a more detailed preregistration.

4 Open Data

In Neuroimaging research, we work with human data. On the one hand, this is the reason why it would be unethical *not* to share it: Because people willingly lend themselves to our experiment, it is our responsibility for the data to have maximum impact. This can only be the case when it is possible to reuse the data, which makes data sharing so important. On the other hand, does the fact that we deal with human data complicate things. People want to stay anonymous, which is why we have to take measures to ensure such anonymity. But although there are some pitfalls when it comes to the sharing of imaging data, there have been efforts to make sharing possible, the results of which I will share in this chapter.

The first prerequisite for sharing imaging data is to get the consent of the participants of the study. The Open Brain Consent Form is a form that is likely to be approved by any ethics committee. After getting consent, the data have to be anonymized. In terms of fMRI data, there is a bunch of tools that can do that automatically, which I will talk about in the section Anonymization. Now that you have consent and an anonymized (and BIDS conform) data set, you can upload it. Where and how to do that, you can read in the section on Data Repositories.

4.1 Open Brain Consent

The Open Brain Consent is a consent form that allows sharing of anonymized imaging data. Currently, two versions exist: A single access type version and a version with two access types. In the single access type version, all data are shared publicly. The other version only allows for some data to be shared publicly, but with the possibility of more to be shared with approved researchers.

While I am writing this, the form is available in English, German, French, Chinese, Spanish, and Italian. You can find the form here.

4.2 Anonymization

In order to share imaging data, they have to be anonymized. While for behavioral data it is only necessary to exclude any personal information from the data set, the case proves a bit more complicated for imaging data.

The first step is to exclude any personal information from your DICOM files. There are two methods which can be used black and whitelisting. Blacklisting means that you indicate all things that you *do not* want published. Whitelisting on the other hand means that you specify the things you *do* want published. Either way can be done using PyDICOM's deid. You can install it as a Docker container or locally. Both ways are described here.

By reconstructing the face from an MRI scan, a third party could theoretically identify the person from the imaging data alone. Stripping away the part of the image that contains the face can prevent this. While there are a bunch of options which are described on the page of the Open Brain Consent, the easiest is probably to use BIDSonym. It is a BIDS app that interfaces several defacing algorithms. If you don't know which to choose, stick with pydeface, as it is recommended by data repositories such as OpenNEURO. Being a BIDS app, it requires your data to already be in the BIDS format, so make sure to convert your data before proceeding.

4.3 Data Repositories

Now that you have BIDS formatted and anonymized data, it is time to upload them. On OpenNEURO, you can upload not only MRI data, but also data acquired using EEG, MEG, and more. If your data are not in the right format, or you have other data than imaging data you want to share, you can also do so on OSF. The Open Science platform offers unlimited space and the data will be directly linked to your project.

5 Open Access

One of the central goals of Open Science is to make research accessible for everyone. After all, it is often funded by public money. Due to this, the public should also benefit from it, which is only possible if the results of scientific studies are openly accessible. This thought has led to the rise of Open Access journals. There are two kinds of Open Access: gold Open Access and green Open Access.

When publishing gold Open Access, readers can read the final, peer-reviewed version of the article for free. This has one drawback: the author usually has to carry the cost of publication, though there are some free Open Access journals. The page Eigenfactor.org lists these. Additionally, it tells you about the cost effectiveness of a given journal, a measure that takes into account the publication cost in relation to the article influence. Another thing could also help you publish in an Open Access journal: some universities have a fund specifically for the costs of Open Access publications. Also, if you still do not have enough money, most Open Access journals will offer to waive part of the fee upon request.

In green Open Access, it is not possible to access the final version for free, or at least not right away. It is however allowed to archive a preprint: the version of the article that is submitted to the journal, but before peer-review. Today, most journals allow archiving preprints. In the next section, I will go more into detail on the practicalities of preprints.

5.1 Preprints

Although the best option is to publish your study in an Open Access journal, this might not always be possible. Then it is a good alternative to archive a preprint. But where should you upload it to?

The first two options are bioRxiv or PsyArXiv. These are dedicated preprint servers for biological and psychological papers. Especially bioRxiv seems to have importance in the neurosciences (“bioRxiv vs. PeerJ Preprints”, n.d.; link), in contrast to the direct competitor PeerJ Preprints. Regardless of your choice, it is possible to link the preprint to your project on OSF for better visibility. You can also directly upload your preprint on OSF.

Another option is to upload the preprint on Reseachgate, a social network for researchers. Alternatively, you can link to a preprint that you uploaded on one of the preprint servers. This is also a good way to get feedback on your article and might enable collaboration on future projects.

One thing to make sure is if your journal allows the archiving of preprints. Likely, this will be the case, but there are still some exceptions. The website SHERPA/RoMEO offers a quick check regarding the policies of journals on self-archiving. Just type the name of the journal and you will have your answer shortly.

6 Concluding Remarks

Originally, the idea was to end up with a kind of workflow of how to conduct a study as openly and transparent as possible. However, given the vast amount of options in every step, this proved difficult in practice. Still, I hope that with the introduction I gave to each of the tools, you are able to mix and match, adjusting the selection of tools to your needs.

The amount of things to do might be a bit overwhelming at first. The thing to remember is that you don't have to apply everything at once. For example, publishing in an Open Access journal will already have a big impact in terms of openness, without much extra effort. Just start with whatever seems easy to you and apply it to your next research project.

If you want to go all in and make all your analysis pipelines reproducible, you might ask yourself what to learn first. I think a good place to start is working through the Nipype tutorial. While of course Nipype stands at the center of this, it provides a good introduction to many other aspects of reproducible analysis.

Another important thing to know is how to get help. Googling things will help you with most problems, though sometimes it might take a while until you get things to work the way you want them to. If you get completely stuck, don't give up! The Neurostars forum is a good place to ask all kinds of questions regarding the reproducible analysis of imaging data. It is the place where the people hang out that wrote many of the programs that I described in this guide—so you are in good hands.

I hope this guide gave you a good place to start out in the world of open science, and that it will help you doing open and reproducible research on your own!

Literature

5 types of software licenses you need to understand | Synopsys. (2016). Retrieved August 2, 2019, from Synopsys website: https://www.synopsys.com/blogs/software-security/5-types-of-software-licenses-you-need-to-understand/

AsPredicted: Home. (n.d.). Retrieved September 9, 2019, from https://aspredicted.org/index.php

Begley C. Glenn, & Ioannidis John P.A. (2015). Reproducibility in Science. *Circulation Research*, *116*(1), 116–126. https://doi.org/10.1161/CIRCRESAHA.114.303819

bioRxiv vs. PeerJ Preprints. (n.d.). Retrieved September 18, 2019, from http://www.omnesres.com/blog/biorxiv_vs_peerj/

Brain Imaging Data Structure. (n.d.). Retrieved July 31, 2019, from https://bids.neuroimaging.io/

Kasberger, S. (2013). Open Science. Retrieved July 23, 2019, from OpenscienceASAP website: http://openscienceasap.org/open-science/

Kraker, P., Leony, D., Reinhardt, W., & Beham, G. (2011). The case for an open science in technology enhanced learning. *International Journal of Technology Enhanced Learning*, *3*(6), 643. https://doi.org/10.1504/IJTEL.2011.045454

McKiernan, E. C., Bourne, P. E., Brown, C. T., Buck, S., Kenall, A., Lin, J., … Yarkoni, T. (2016). How open science helps researchers succeed. ELife, 5, e16800. https://doi.org/10.7554/eLife.16800

Nuzzo, R. (2015). How scientists fool themselves – and how they can stop. *Nature News*, *526*(7572), 182. https://doi.org/10.1038/526182a

Open-notebook science. (2019). In *Wikipedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Open-notebook_science&oldid=902373988

Open Science Collaboration. (2015). PSYCHOLOGY. Estimating the reproducibility of psychological science. *Science (New York, N.Y.)*, *349*(6251), aac4716. https://doi.org/10.1126/science.aac4716

Rosenthal, R. (1979). The file drawer problem and tolerance for null results. *Psychological Bulletin, 86*(3), 638-641. http://dx.doi.org/10.1037/0033-2909.86.3.638