Making Reproducible Research Simple Using RMarkdown and the OSF

André Calero Valdez(✉)

Human-Computer Interaction Center, RWTH Aachen University,
Campus Boulevard 57, 52076 Aachen, Germany
calero-valdez@comm.rwth-aachen.de

Abstract. The replication crisis has further eroded the public's trust in science. Many famous studies, even published in renowned journals, fail to produce the same results when replicated by other researchers. While this is the outcome of several problems in research, one aspect has gotten critical attention—reproducibility. The term reproducible research refers to studies that contain all materials necessary to reproduce the scientific results by other researchers. This allows others to identify flaws in calculations and improve scientific rigor. In this paper, we show a workflow for reproducible research using the R language and a set of additional packages and tools that simplify a reproducible research procedure.

Keywords: Reproducible research · Replication crisis · Literate programming

1 Introduction

The scientific database Scopus lists over 73,000 entries for the search term "reproducible research" at the time of writing this document. The importance of making research reproducible was recognized in the early 1950s in multiple research subjects. And with the reproducibility project, the Open Science Foundation [26] found that merely half of all studies conducted in psychological research can be replicated by other researchers. Several factors have contributed to this problem. From a high-level perspective, the pressure to publish and the increase in scientific output has lead to a plethora of findings that will not replicate. Both bad research design and (possibly unintentional) bad research practices have increased the number of papers that hold little to no value. More than half of researchers agree that there is a severe reproducibility crisis in science according to Baker [3] and her article in Nature. The study also found that problems for reproducibility include: a lack of analysis code availability, a lack of raw data availability, and problems with reproduction efforts.

2 Problematic Research Practices

One problem that is often mentioned is HARKing [16] or "hypothesizing after results are known". When multiple statistical tests are conducted with a normal alpha-error rate (e.g., $\alpha = .05$), it is expected that some tests will reject the null-hypothesis on mere randomness alone. Hence, the error-rate. If researchers now claim that these findings were their initial hypotheses, results will be indiscernible from randomness. However, this is unknown to the reviewer or reader who only hears about the new hypotheses. HARKing produces findings were there are none. It is thus crucial to determine the research hypothesis before collecting (or analyzing) the data.

Another strategy applied (often without ill intent) is p-hacking [13]. This technique is widespread in scientific publications and probably already is shifting consensus in science. p-hacking refers to techniques that alter the data until the desired p-value is reached. Omitting individual *outliers*, creating different grouping variables, adding or removing control variables—all these techniques can be considered p-hacking. This process also leads to results that will not hold under replication. It is crucial to show what modifications have been performed on data to evaluate the interpretability of p-values.

When researchers already "massage" the data to attain *better* p-values, it is additionally bad that many researchers do not understand the meaning of p-values. As Colquhoun [9] found, many researchers misinterpret p-values and thus frame their findings much stronger than they really are. Adequate reporting of p-values is thus important to the interpretability of results as well.

Lastly, scientific journals have the problem that they are mostly interested in publishing significant results. Thus contradictory "non-findings" seldom get published in renowned journals. There is little "value" for a researcher to publish non-significant findings, as the additional work to write a manuscript for something like *arXiv* does often not reap the same reward as a journal publication. This so-called *publication bias* [29] worsens the crisis. As now only significant findings are available. It is thus necessary to simplify the process of publishing non-significant results.

3 Reproducible Research Workflows

Many different solutions to this process have been proposed to address these challenges (e.g., [22,37]). However, no uniform process exists that allows the creating of documents and alternative reproducibility materials in one workflow.

In this paper, we demonstrate a research workflow based on the R-language and the R Markdown format. This paper was written using this workflow and the sources are freely available online (https://www.osf.io/kcbj5). Our workflow directly addresses the challenge of writing LNCS papers and a companion paper website (https://sumidu.github.io/reproducibleR/) that includes additional material and downloadable data.

In this paper, we will focus on the following aspects:

- Creating a reproducible research compendium using RMarkdown
- Using GitHub and the OSF to make research accessible
- Packages that simplify research in RStudio

We assume that the reader is somewhat familiar with the R Programming language and knows that scientific analyses can be run using computational tools such as R, Python, Julia or others. The guidance in this paper addresses the R user.

3.1 What Is Reproducibility?

The Open Science Foundation (OSF) speaks of three different kinds of reproducibility [24]. *Computational reproducibility* refers to the quality of research that when other researchers get access to your code and data that they will be able to reproduce your results. *Empirical reproducibility* means that your research has sufficient information that allows other researchers to recreate your experiments and copy your study. *Replicability* refers to the quality of an outcome and a study, meaning that given that you were to reproduce the experiment, you would also reach the same outcome. In this article, we provide tools for the first type of reproducibility only, as the latter are both dependent on your research content not exclusively on your procedure. It is important to note that creating computationally reproducible research is important, but it is also worthless when basic concepts of methods and research processes are ignored. If you measure incorrectly, your result may reproduce, but the finding may be wrong anyways. Hopefully, when you are using the suggested workflow here, others will be able to point out mistakes to you more easily.

4 Writing a Research Compendium

The central aim of a research compendium is to provide all data and information necessary to allow others to reproduce your findings from your data [12]. There are several different ways of achieving this but a central theme of a research compendium is to organize data in a meaningful fashion. Since we are addressing R users, it makes sense to consider possible computing environments for R first.

You can find detailed information on how to create a research compendium online here https://research-compendium.science/.

4.1 Why R and RMarkdown?

R is the de-facto standard when it comes to statistical analysis tools that are open source and free to use. In economics and the social sciences, similar tools that provide a GUI like SPSS are used with one immediate downside for reproducibility. If your analysis toolkit is proprietary, other users will not be able to reproduce your work without a significant investment.

Moreover, using a GUI makes it untraceable—even to yourself—what analyses you have conducted later. You might have manually deleted a row with broken data, or might have recoded a typing error in your data manually. If this is not documented, this information is lost. Using a language like R, where every change of the data corresponds to a line of code, no accidental "quick fixes" will get lost over time. R also provides a rich set of tools for reproducible research on CRAN[1].

4.2 Literate Programming

RMarkdown is a tool that is extremely helpful for researchers, as it allows us to combine analysis code with regular text. This document was written using RMarkdown and integrating some analysis code in between. RMarkdown is a *literate programming* approach. The documentation of code is equally necessary for understanding the code, as the code itself. By interleaving code and text, the intentions of the developer are implicitly communicated. Python and Julia have similar approaches by using Jupyter notebooks.

RMarkdown allows not only for the integration of text and figures directly from code, but it also allows writing in an abstract format. A single document (such as this) can be rendered to various output formats. In this case, it is rendered to the LNCS styled Latex output format, as well as to a website using bootstrap. The benefit is that text and code are reusable, so when papers get rejected no excessive reformatting has to be made. Formatting is done using Markdown (see here[2] for a tutorial). Code and analyes are interleaved in text in so-called "code chunks". Code chunks can contain R code, but also code from other languages (e.g., Python).

4.3 Project Workflows

The most popular integrated development environment (IDE) for R is RStudio. RStudio comes with a license that allows researchers to freely use it for scientific purposes and it integrates many of the tools described in this paper. The first strong tool for reproducible research using R is using RStudio projects.

RStudio projects contain information about where your code, your data, and your output should reside on your computer. The benefit of RStudio projects is that they contain relative path information, so when another user installs your project on their computer, it should work without a problem. Since you need to refer to files in some cases, even relative paths work well. The `here` package provides a helpful tool to access data relative to the project main directory. This works on Linux, Windows, and Mac computers.

4.4 Package Management

Another key requirement for computational reproducibility is that the software versions on different computers actually produce the same analysis. This is most safely achieved by keeping all libraries in the same version as in the original analysis. Sometimes libraries change their features and this can render old projects

[1] https://cran.r-project.org/web/views/ReproducibleResearch.html.
[2] https://www.markdowntutorial.com/.

unusable. Using package management is not only a necessity for computational reproducibility, but it is also helpful for yourself when you get back at a project. There are several tools in the R universe that address this challenge. All of them have different efforts involved and provide different benefits.

The **packrat** package [32] comes integrated into RStudio and allows you to create a localized copy of the used libraries in your analysis. Packrat even downloads sources of these packages and allows using libraries from different sources (CRAN, GitHub, etc.). Packrat provides a function (`packrat::bundle()`) to pack everything into a shareable file. However, packrat sometimes has problems with multiple RMarkdown files in the project, causing it to re-render all documents to infer the used packages. In these cases packrat is not a viable solution. When you find your project becoming very slow, it might make sense to remove packrat.

This is where the **renv** package [31] comes into play. It is a simplified version of package management and runs relatively reliable even with multiple RMarkdown files in the project. By calling `renv::init()` a lock file is created that contains information on all packages used in the project. It does, however, not download sources, so it will only work if the packages you use are expected to be available in the future as well.

Neither of these options though addresses the challenge of using the same R-version or the same operating system. Differences between Windows and Linux could yield different results in the future. This is where **docker** comes into play. Docker is a light-weight virtualization software that allows you to run a virtual machine based on other users' machine images. It also provides a sharing platform for these images. Rocker provides a set of default virtual machine images[3] that contain both a fixed R version and a set of libraries usable in research. By adding a `dockerfile` to a project, you can create a definition of your project that will build a matching machine image. Docker does require the user to install the docker software on their machine.

There are options for sharing your run-time environment without asking other researchers to install any software. RStudio comes with a cloud version that can (currently) be used for free if the projects are either private or completely public. By running your project in `rstudio.cloud` and sharing the public link to a project, others can create copies of your run-time environment in their **rstudio.cloud** account.

An even simpler version is the use of **binder**. Binder can be set up to automatically build your project on a virtual machine and provide an RStudio instance on the virtual machine that has access to your project. However, to make this work, you need to use a version control system, more specifically you need to use GitHub with your project. The auto-generated binder link from the www.mybinder.org website can be extended to use RStudio by adding `?urlpath=rstudio` to the URL. You can have a lookt at the readme of this project on GitHub to see how it is done.

[3] https://github.com/rocker-org/rocker.

4.5 Writing Articles Using `rmdtemplates`

The package `rmdtemplates` [7] provides templates for writing RMarkdown files that adhere to the Lecture Notes in Computer Sciences series. It also contains a template for an open data website. This website allows creating a reader-friendly version of your paper to send around. Moreover, you can add additional analyses in the open data website which can then be added as supplementary materials to your paper. You must ensure though that copyrights are respected when sharing the written content of your paper.

Key benefits of the `rmdtemplates` package are that it supports to automatically generate citations for the R packages that you use. RMarkdown uses bib-files to store your references. To make referencing easier, install the `citr` [2] package in RStudio to enable a GUI to use references from your library. Citr allows connecting your Zotero database and using those references as well.

5 Open Data and Open Code

One key idea of reproducibility is making data and analysis code openly available. Sharing your code allows other researchers to inspect it and verify that your results are valid conclusions from your analyses. Research and statistical analyses are complex processes and mistakes are bound to happen sometime. Mistakes are less severe when they can be retraced and results adapted. Sharing your data allows other researchers to see what other information might have been undiscovered by your analyses. Studying open data can be used to explore new theories, used in meta-analyses and be used in teaching settings. Typically data is released under the CC0 license, making data part of the public domain.

5.1 Data Sharing and Anonymization

Sharing data is not just uploading your data to a website. First, several considerations must be made before data can be shared. The most important question is: "Does my data contain personal information?" Any data that was collected on human subjects potentially contains personal information. This has several implications.

First, you must ask whether the participants agreed with data sharing. Typically, participants sign data waivers allowing researchers to use data for scientific purposes. It is important to inform participants about the possibilities of sharing.

Second, information on people can be damaging to these people upon release. By allowing others to utilize your data, you must consider possible threats to your participants before deciding what data to release. It is crucial to inform yourself about data and anonymization before carelessly releasing information. For example, releasing information on your participants when they were students from a certain semester might leave individuals identifiable in your data set. Thus it may be necessary to either anonymize your data or to limit additional information on data gathering procedures (or both). It makes sense to speak to an expert on anonymization about this topic and to ask for permission from your organization's ethics board.

K-Anonymity. The simplest form of anonymity can be generated if all quasi-identifiers of a person appear mulitple times in the database from several other persons. Each person is then represented by the same data attributes so that each person can no longer be distinguished from other persons with the same attributes. This concept is called k anonymity [1]. If at least k persons exist in a given data set, who are identically represented in terms of their quasi-identifiers, the data is k-anonymized. Each person is now in an *equivalence class* of at least $k - 1$ other people who share the same *quasi-identifiers*.

This method provides an intuitive version of privacy that is both algorithmically simple to implement and easy to explain to the participant. Technically, we can simply add noise to the data to enhance privacy. For example, we can remove the last digits of postal codes (data deletion) until at least k equal entries exist for each postal code. We can also store age groups instead of birth dates (data aggregation).

The advantage of this method is that our data is only slightly changed, as only the quality of the data is reduced. One problem is not solved with this method. It could be that the combination of quasi-identifiers and sensitive data could still be too informative for an external attacker. An insurance company might want to know that all persons from a region aged 65 and older suffer from heart disease. Even if no person is de-anonymized in this scenario, all persons in the data set may suffer from the consequences of possible secondary use of the data.

The most important finding of k-anonymity is that the most important problem in anonymization is not user identification, but data sensitivity and the possibilities of secondary use. For this purpose l-diversity [20] or t-closeness [19] may be considered. A package for R that provides an interactive tool for applying anonymization techniques to a data set is the `sdcMicro` [30] package. Another option is the `anonymizer` [14] package which provides methods for detecting potentially identifying information and replacing it with hashes.

Differential Privacy. Often other scientists are not interested in individual data. If only the statistical properties of a data set are interesting, it should be easy to ensure the privacy of individuals. However, it is still possible to obtain sensitive information about individual users by repeatedly querying a database.

Attackers can combine multiple queries to narrow down sensitive information about individuals. The idea behind Differential Privacy is to establish a privacy budget [11]. Whenever statistics are calculated on the data, the amount of information in these statistics is deducted from this privacy budget. This is achieved by replacing data with noisy data. This means that two identical database queries will most likely yield different results. The more queries are received, the more different the results become until the database returns only noise. The database must now either be discarded or new data must be collected to increase the budget for data protection.

The advantage of Differential Privacy is that there is a mathematically guaranteed privacy for each user. It is therefore impossible to gain knowledge about individuals from the retrieved information [18].

5.2 GitHub and Git

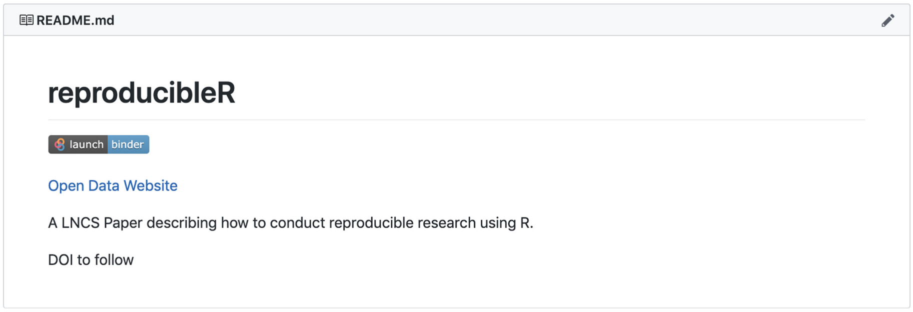
Sharing of code is a procedure that is natural to computers scientists, as almost all larger software products are team efforts. GitHub has crystallized as the de-facto standard of sharing code for open-source software. What is GitHub and how do you use it?

First, we must understand **Git**. Git is a version control software (VCS). Git allows you to keep track of changes in your files. It allows you to store individual changes as so-called *commits*. Each individual commit can always be restored from the git *repository* on your computer. This gets read of the challenge of keeping multiple version files of a document. Git works completely locally, so you can move project folders that are tracked by git around on your computer or someone elses computer, without losing tracking information.
RStudio is completely integrated with Git, so committing new versions of your project is as simple as a click. Git has proven to be the most valuable tool in literate programming for science [6].

GitHub is a website that provides free repositories for open source software or open-source research. GitHub allows you and your collaborators to work on the same project asynchronously. By uploading (called *pushing*) your local git repository to the public GitHub repository your collaborators or other researchers get access to this project. These people can now download the repository (called *pulling*) to their computer and work on the project or reproduce your analysis. Git has extensive mechanisms for merging your progress and your collaborator's project progress. Changes can be integrated on a line-by-line basis. Thus it is best to break lines in your code frequently.

It is important to note that GitHub is not the best place to store your data. Individual files are limited to 100 MB and projects are limited to 2 GB.

GitHub Readme. Beyond providing a publicly available place to store your analysis code. GitHub serves as a publicly accessible website for your research project. It is recommended to upload a `README.md` file that contains basic information about your research project. It could contain a DOI of the published article, it could contain links to other parts of the project such as data stores on the web. The benefit of GitHub readme files is that they will automatically render a pretty HTML output on the website (see Fig. 1).


README.md

reproducibleR

Open Data Website

A LNCS Paper describing how to conduct reproducible research using R.

DOI to follow

Fig. 1. Rendered Readme.MD file on GitHub.

GitHub Pages. If you have generated your analysis using RMarkdown you can render your output to a website as well. This provides the benefit of adding additional figures and making your document more accessible. By using libraries such as `plotly` other researchers can even explore your data using interactive visualizations. The template from the `rmdtemplates` [7] package provides a nice pre-structured interactive website that allows you to include tabular downloadable data in the website.

When you store your projects on GitHub, you can make your website publicly available easily. By copying the output format to a sub-folder called `docs` and enabling GitHub Pages in your GitHub settings your page is exposed to the public without requiring a hosting service (except for GitHub).

5.3 OSF

While GitHub is an excellent provider for storing the code of your analysis, it is not very well suited for sharing data and for reviewing purposes. The Open Science Foundation (OSF) provides a service where researchers can create projects that have Wikis, file storage, and transparent referencing. You may even choose the server where your data is stored when data protection laws require your data to be in your country.

A key benefit of the OSF is that you can create sub-modules in your project and share the whole project or the sub-modules individually with others. Each "node" in your project gets an easy-to-recognize and short unique URL which can be added to a paper or a website. More importantly, it allows the sharing of parts of your project anonymously, during the reviewing process. The reviewers can see the available data, without seeing the authors' names. But they also can verify that data has not been changed since the project has gone public.

To make things even better, there is a package called `osfr` [38] that lets you download (or upload) your data directly from your R or RMarkdown documents. This can be leveraged in the setup procedures of a document, as in: "If the data is not available, try downloading it automatically."

Preregistration. Another benefit of the OSF is that you may preregister your study before collecting data. By setting up preregistration at the OSF and setting up the rest of your project before collecting data, you prevent yourself from HARKing in your research. Preregistered trials are part of the gold standard of high-quality social science research. Some journals have decided to accept studies after preregistration to prevent a publication bias towards significant findings. This does not mean that you can no longer conduct exploratory research on your data, it simply ensures that confirmatory and exploratory findings are clearly separated.

6 Helpful Tools for Everyday Tasks

In this section, we will introduce some tools that make life as a researcher easier, when relying on computational analysis using R and RStudio.

6.1 Automizing Builds Using Drake

One large challenge for literate programming is that for every small change in a document you need to re-run your complete analysis. This is not a problem with modern computers in many cases, but when your data is large enough, it can become a hassle. While RMarkdown does provide a caching mechanism, it is limited to the individual computer and may not be shared among researchers. And when individual code chunks depend on external data that has changed, RMarkdown caching no longer registers these changes.

The package `drake` [17] addresses this challenge. By creating a plan using the `drake_plan` function we first determine what steps are necessary for our analysis. Drake then analyzes our code and files for implicit dependencies. It derives a dependency tree (see Fig. 2) that visually shows how the project should run.

Each build target (e.g., `report`) is the call to an R function. The result of which can easily be reloaded anywhere inside the project using the `loadd` or `readd` function. In the simplistic build path for this document, the `hist` target creates a histogram of the `iris_data`, which is downloaded from the OSF. The figure (see Fig. 3) is then included in the document. Whenever a single previous dependency becomes outdated, the rest of the dependency graph is executed.

When all dependencies are properly modeled, drake allows running individual targets on multiple CPUs or a compute cluster without much overhead. The interested reader can take a look at the `make.R` file in the project root of this document.

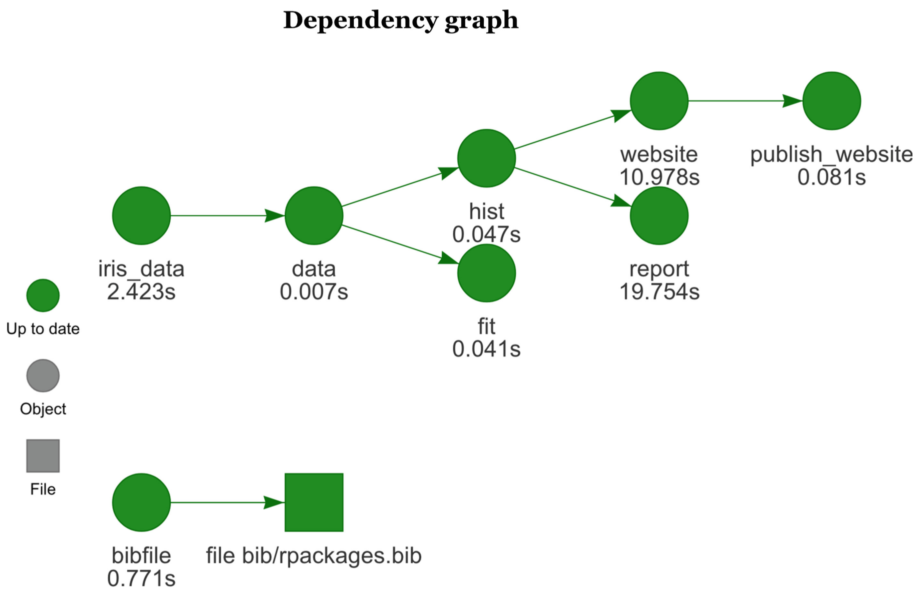


Fig. 2. The drake plan to generate this document.

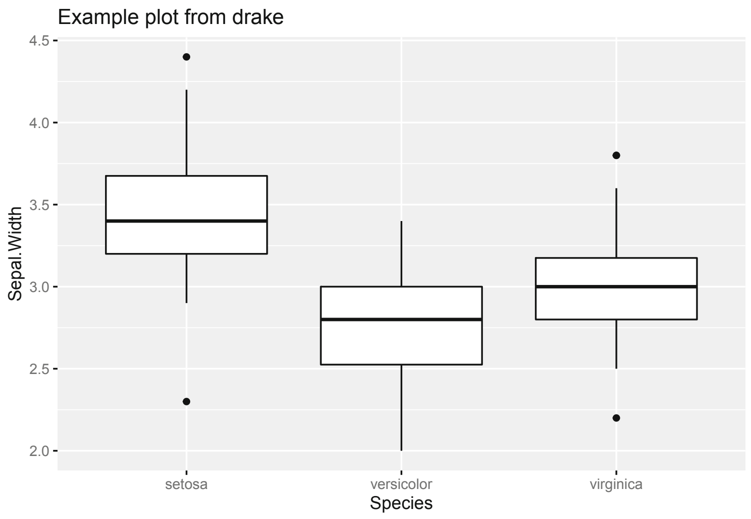


Fig. 3. This figure was created outside of the document in the hist target.

6.2 Ensuring Relative Paths Using `here`

The `here` [25] package provides a useful tool to find files on all operating systems. Windows and Unix are famous for using different slashes as directory separators, which can make addressing file paths complicated for reproducible research. By encapsulating all file operations in the `here` function, the relative root is set to the destination of the R project file and directory separators are automatically inserted in accordance with the local operating system.

6.3 Automating Project Setup Using `usethis`

When setting up a project, many tasks have to conducted over and over again. To simplify this the `usethis` [35] package provides a set of tools to start working on a project. A typical workflow for setting up a project using `usethis` could look like this.

1. Create a project using `create_tidy_project`
2. Setup the license using, e.g., `use_mit_license`
3. Setup using git `use_git`
4. Setup using GitHub `use_github`
5. Setup a readme using `use_readme_rmd`
6. Setup a citation for your project `use_citation`

Running `git_vaccinate` once will add all files that typically contain credentials or other personal information to the global git-ignore file, preventing them from being accidentally shared.

Another library that helps with setting up a project is the `rrtools` [21] package.

6.4 Onboarding New Users to RStudio with Addins

Learning how to write code can be hard for someone changing over from UI-based approaches such as SPSS. However, there are several tools that help you create reproducible R code from the UI of RStudio. Such tools can be found in the “Addins” menu and we will highlight some that make research easier.

Working with factors is not always easy in R. They will appear in alphabetic order in plots, they are hard to rename and hard to reorganize. The `forcats` [33] package simplifies the use of factors, by unifying the interface to them. An addin from the `questionr` [5] package allows for easy recoding and relabeling of factors.

Creating custom plots using the `ggplot2` package creates usable figures for scientific papers. Yet, it may take a while to get accustomed to ggplot. The `esquisse` [23] package has an interactive addin (see Fig. 4) that lets you drag and drop variables to axes, adjust layout and color, filter the data, and export the script that would generate the matching plot.

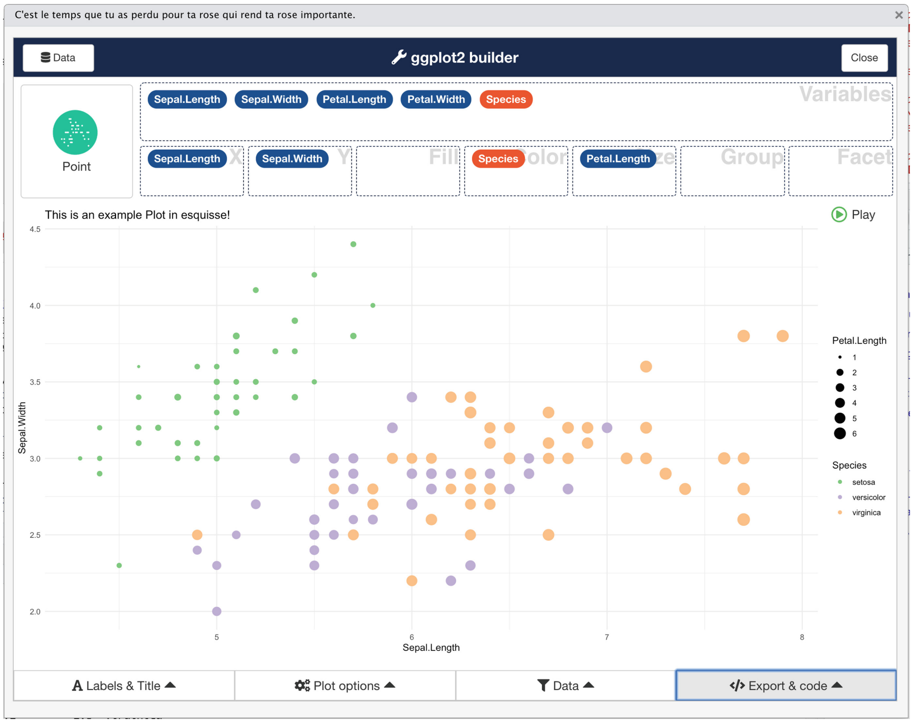


Fig. 4. This is the interface to the ggplot builder from esquisse.

6.5 Using `ggstatsplot` to Generate Meaningful Statistical Analyses

Knowing how to report a statistical finding often requires a deep understanding of what test to use and how to report the results adequately. The `ggstatsplot` [27] package provides a set of plotting functions that use very sensible defaults derived from the data input. If, for example, you want to compare means between multiple groups the `ggbetweenstats` function will produce a nice plot with all relevant statistical information—including effects sizes, confidence intervals, and Bayes factors.

The upcoming plot (Fig. 5) was created from a single line of code.

```r
ggbetweenstats(iris, Species, Sepal.Width, messages = F)
```

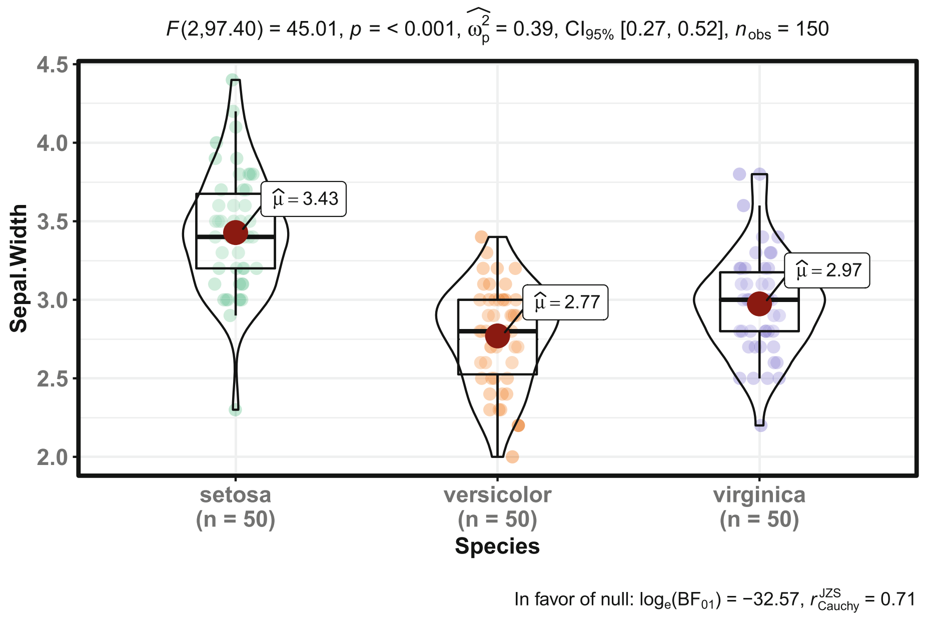


Fig. 5. A comparison of means using the iris data.

6.6 Create Research Plans Using `DiagrammeR`

Even process diagrams as in Fig. 6 can easily be created using the `DiagrammeR` [15] package. It requires writing a process description in the `dot` language which is relatively easy to learn.

```
library(DiagrammeR)

grViz(diagram = "
      digraph boxes_and_cicrles {

      graph [rankdir = TB]

      node [shape = box
            fontname = Helvetica
            ]
      'Setup OSF Project Site'
      'Setup R Project'
      'Setup GitHub Repo'
      'Ensure reproducibility using renv'
      'Write analysis'
      'Preregister Study'
      'Collect Data'
      node [shape = circle]

      Start
      'Submit Paper'

      edge []

      Start->'Setup OSF Project Site';
      'Setup OSF Project Site'->'Setup R Project';
      'Setup R Project'->'Setup GitHub Repo';
      'Setup GitHub Repo'->'Ensure reproducibility using renv';
      'Ensure reproducibility using renv'->'Write analysis';
      'Write analysis'->'Preregister Study';
      'Preregister Study'->'Collect Data';
      'Collect Data'->'Submit Paper'
      }
      ")
```

7 Discussion

In this paper, we have very shortly introduced a large number of tools that can be utilized to make research more computationally reproducible. By integrating the different tools into a complete workflow, emergent effects from the interactions of the individual steps can be reaped. However, this workflow can easily become overwhelming as it includes multiple tools, each of which has large documents attached to them explaining how to use them. You as a reader may decide on which tools to use from this set and which to ignore. Be warned though, that skipping some of the steps will reduce the benefits for other researchers and yourself.

We did not extensively elaborate on how some of these tools are used most effectively. We leave it up to you to deepen the knowledge of some of these tools. In the future, we want to create additional tutorial materials made available on the website for this paper[4].

[4] (https://sumidu.github.io/reproducibleR/).

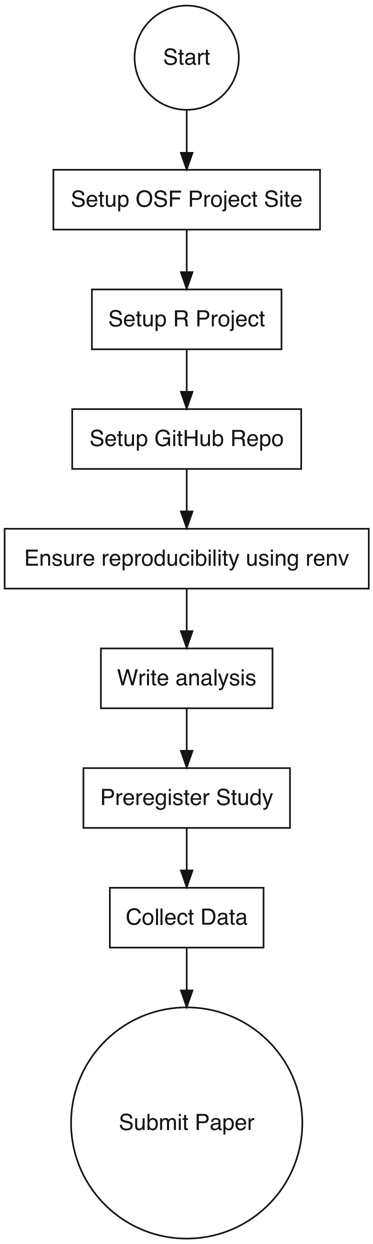


Fig. 6. Reproducible workflow using the tools from this paper.

Acknowledgements. This research was supported by the Digital Society research program funded by the Ministry of Culture and Science of the German State of North Rhine-Westphalia. We would further like to thank the authors of the

packages we have used. We used the following packages to create this document: `knitr` [39], `tidyverse` [34], `rmdformats` [4], `kableExtra` [40], `scales` [36], `psych` [28], `rmdtemplates` [7], `sdcMicro` [30], `webshot` [8], `here` [25], `DiagrammeR` [15], `citr` [2], `drake` [17], `esquisse` [23], `usethis` [35], `gramr` [10], `questionr` [5], `ggstatsplot` [27].

References

1. Aggarwal, C.C., Philip, S.Y.: A general survey of privacy-preserving data mining models and algorithms. In: Aggarwal, C.C., Yu, P.S. (eds.) Privacy-preserving data mining, vol. 34, pp. 11–52. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-70992-5_2
2. Aust, F.: citr: RStudio Add-in to Insert Markdown Citations. R package version 0.3.2. (2019). https://CRAN.R-project.org/package=citr
3. Baker, M.: Reproducibility crisis. Nature **533**(26), 353–66 (2016)
4. Barnier, J.: rmdformats: HTML Output Formats and Templates for 'rmarkdown' Documents. R package version 0.3.6. (2019). https://CRAN.R-project.org/package=rmdformats
5. Barnier, J., Briatte, F., Larmarange, J.: questionr: Functions to Make Surveys Processing Easier. R package version 0.7.0. (2018). https://CRAN.R-roject.org/package=questionr
6. Bryan, J.: Excuse me, do you have a moment to talk about version control? Am. Stat. **72**(1), 20–27 (2018)
7. Valdez, A.C.: rmdtemplates: rmdtemplates - an opinionated collection of rmarkdown templates. R package version 0.4.0.0000. (2020). https://github.com/statisticsforsocialscience/rmd_templates
8. Chang, W.: webshot: Take Screenshots of Web Pages. R package version 0.5.2. (2019). https://CRAN.R-project.org/package=webshot
9. Colquhoun, D.: The reproducibility of research and the misinterpretation of p-values. Roy. Soc. Open Sci. **4**(12), 171085 (2017)
10. Dumas, J., Marwick, B., Shotwell, G.: gramr: The Grammar of Grammar. R package version 0.0.0.9000. (2020). https://github.com/ropenscilabs/gramr
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
12. Gentleman, R., Lang, D.T.: Statistical analyses and reproducible research. J. Comput. Graph. Stat. **16**(1), 1–23 (2007)
13. Head, M.L., et al.: The extent and consequences of p-hacking in science. PLoS Biol. **13**(3), e1002106 (2015)
14. Hendricks, P.: anonymizer: Anonymize data containing personally identifiable information. R package version 0.2.2. (2020). https://github.com/paulhendricks/anonymizer
15. Iannone, R.: DiagrammeR: Graph/Network Visualization. R package version 1.1.0. (2020). https://github.com/rich-iannone/DiagrammeR
16. Kerr, N.L.: HARKing: hypothesizing after the results are known. Pers. Soc. Psychol. Rev. **2**(3), 196–217 (1998)
17. Landau, W.M.: drake: A Pipeline Toolkit for Reproducible Computation at Scale. R package version 7.10.0. (2020). https://CRAN.Rproject.org/package=drake
18. Lee, J., Clifton, C.: How much is enough? choosing e for differential privacy. Inf. Secur. **7001**, 325–340 (2011)

19. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 106–115. IEEE (2007)
20. Machanavajjhala, A., et al.: l-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1), 3 (2007)
21. Marwick, B.: rrtools: Creates a Reproducible Research Compendium. R package version 0.1.0. (2019). https://github.com/benmarwick/rrtools
22. Marwick, B., Boettiger, C., Mullen, L.: Packaging data analytical work reproducibly using R (and friends). Am. Stat. **72**(1), 80–88 (2018)
23. Meyerm, F., Perrier, V.: esquisse: Explore and Visualize Your Data Interactively. R package version 0.3.0. (2020). https://CRAN.Rproject.org/package=esquisse
24. Meyers, N.K.: Reproducible Research and the Open Science Framework (2017). https://osf.io/458u9/
25. Müller, K.: here: A Simpler Way to Find Your Files. R package version 0.1. (2017). https://CRAN.R-project.org/package=here
26. Open Science Collaboration et al.: Estimating the reproducibility of psychological science. Science **349**(6251), aac4716 (2015)
27. Patil, I.: ggstatsplot: "ggplot2" Based Plots with Statistical Details. R package version 0.2.0. (2020). https://CRAN.R-project.org/package=ggstatsplot
28. Revelle, W.: psych: Procedures for Psychological, Psychometric, and Personality Research. R package version 1.9.12.31. (2020). https://CRAN.R-project.org/package=psych
29. Simonsohn, U., Nelson, L.D., Simmons, J.P.: p-curve and effect size: correcting for publication bias using only significant results. Perspect. Psychol. Sci. **9**(6), 666–681 (2014)
30. Templ, M., Meindl, B., Kowarik, A.: sdcMicro: Statistical Disclosure Control Methods for Anonymization of Data and Risk Estimation. R package version 5.5.1. (2020). https://CRAN.Rproject.org/package=sdcMicro
31. Ushey, K.: renv: Project Environments. R package version 0.9.3-30. (2020). https://rstudio.github.io/renv
32. Ushey, K., et al.: packrat: A Dependency Management System for Projects and their R Package Dependencies. R package version 0.5.0. (2018). https://CRAN.R-project.org/package=packrat
33. Wickham, H.: forcats: Tools for Working with Categorical Variables (Factors) (2020). http://forcats.tidyverse.org, https://github.com/tidyverse/forcats
34. Wickham, H.: tidyverse: Easily Install and Load the 'Tidyverse'. R package version 1.3.0. (2019). https://CRAN.R-project.org/package=tidyverse
35. Wickham, H., Bryan, J.: usethis: Automate Package and Project Setup. R package version 1.5.1. (2019). https://CRAN.Rproject.org/package=usethis
36. Wickham, H., Seidel, D.: scales: Scale Functions for Visualization. R package version 1.1.0. (2019). https://CRAN.R-project.org/package=scales
37. Wilson, G., et al.: Good enough practices in scientific computing. PLoS Comput. Biol. **13**(6), e1005510 (2017)
38. Wolen, A., Hartgerink, C.: osfr: Interface to the 'Open Science Framework' ('OSF'). R package version 0.2.8. (2020). https://CRAN.Rproject.org/package=osfr
39. Xie, Y.: knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.28. (2020). https://CRAN.Rproject.org/package=knitr
40. Zhu, H.: kableExtra: Construct Complex Table with 'kable' and Pipe Syntax. R package version 1.1.0. (2019). https://CRAN.R-project.org/package=kableExtra