Amsterdam University Press

COMPUTATIONAL COMMUNICATION RESEARCH ?.? (20??) 1–28
HTTPS://DOI.ORG/10.5117/CCR????

What makes computational communication science (ir)reproducible?

Chung-hong Chan[1]
GESIS – Leibniz-Institut für Sozialwissenschaften

Tim Schatto-Eckrodt
Journalism and Mass Communication, University of Hamburg

Johannes B. Gruber
Department of Communication Science, University of Amsterdam

Abstract
Computational methods are in full swing in communication science. Part of their promise is to make communication research more reproducible. However, how this plays out in practice has not been systematically studied. We verify the reproducibility of the entire cohort of 30 substantive and methods papers published in the journal *Computational Communication Research* (CCR), the official journal of the ICA Computational Methods Division with a focus on transparency and hence a high rate of voluntary Open Science participation in the field. Among these CCR papers, we are not able to verify the computational reproducibility of *16* papers as *no* data and/or code were shared. For the remaining *14* papers, we attempt to execute the code shared by the original authors in a standardized containerized computational environment. We encounter a variety of issues that preclude us from reproducing the original findings, where incomplete sharing of data or code is the most common issue. In the end, we could at least reproduce the findings in *only 6 papers (20%)* partially. We end the paper by discussing strategies for researchers and the subfield to correct for this disheartening state of computational reproducibility.

Keywords: reproducibility, computational communication science, Open Science

Based on various yardsticks, the application of computational methods in communication research is in full swing. Computational communica-

[1]All authors contributed to this project equally. The order of the names was determined by a random draw.

tion science has earned widespread consideration as a subfield of Communication Science (Hilbert et al., 2019). The Computational Methods division of the International Communication Association (ICA CM) is one of the fastest growing divisions of the association: its membership has doubled every one to two years since ICA CM's inception. There is also a steady supply of communication papers using computational methods. According to one estimate, 2% of papers in journalism studies journals in the last decade used a computational approach (Zeng & Chan, 2023). Various traditional communication journals have published special issues devoted solely to computational methods (e.g. *Communication Methods & Measures* 12:2, *International Journal of Communication* 13, and *Political Communication* 38:1). And the most important yardstick—in our opinion—is the dedication of an official ICM CM journal: *Computational Communication Research* (*CCR*).

Two of the promises of computational methods for Communication scholars are that it makes research more transparent and more reproducible. Computer code by design constitutes a detailed set of instructions on how to perform a certain task. Hence, sharing it with other researchers makes the steps taken to arrive at results highly transparent. Additionally, computational methods should make analyses perfectly reproducible, as the same code, running an analysis task on the same data, should always return the same results. At the inauguration in 2019, the founding editors of *CCR* commited to these promises in *"A Roadmap of Computational Communication Research"* (Van Atteveldt, Margolin, et al., 2019, *Roadmap I* hereinafter, emphasis added):

> *"CCR demands transparent and reproducible research. Computational analyses require many choices regarding design, preprocessing, and parameter tuning, and transparency are needed to allow scrutiny of these choices. As digital data and analysis code can be shared easily, computational research can be at the forefront of the open science philosophy [...] Most articles in CCR should be accompanied by an online appendix in a form that encourages reproducibility and reusability. [...] For articles presenting substantive and/or methodological analysis results and data contributions,* ***we expect an online research compendium published on GitHub or an equivalent service. Such a compendium contains the data, code, and results, and makes it explicit how the code is used to derive the results from the raw data."***

However, in practice, sharing code and data **publicly** still comes with obstacles [2] and the code to run an analysis does not only consist of the high-level code generated by the researchers, but comprises many parts: from hardware drivers, low-level operating system features and external dependencies, such as R or Python packages and their dependencies, to randomly generated numbers for sampling and bootstrapping. If code at any stage changes, the efforts to make it possible for others to run analyses again might have been futile or require extended additional efforts. Technically, the computational environment can be controlled by researchers and documented for future reproducibility—if they are aware of issues, technically adapt enough, and willing to invest the effort. If not addressed by the research community, however, we agree with Mede et al. (2020) in worrying that the looming replication crisis could erode public trust in science as a whole.

The demands for transparency and reproducibility in *Roadmap I* therefore deserve our special scrutiny. Not because we doubt the sincerity of the commitment, but because we assume the computational subfield to be ahead of other communication scholars regarding the efforts (and struggles) to make transparency and reproducibility the norm. We assume that communication science mirrors other disciplines here, in which so-called "Computational X" subfields [3] also grapple with reproducibility problems (e.g. Hothorn & Leisch, 2011; Hutson, 2018; Ioannidis et al., 2009). Addressing remaining issues we find today in the official ICA CM journal, which again is probably the most committed to transparency and reproducibility, might thus lay a more steady groundwork for the future of Communication Science and help rectify questionable research practices of the past (Bakker et al., 2021; Matthes et al., 2015). Our goal in this article is thus not to criticize the efforts already made, but to highlight which problems remain in the commendable efforts already undertaken to battle *irreproducibility* of research findings.

This article fist presents an overview of important concepts and research connected to transparency and reproducibility. It then describes the inclusion criteria for articles in our reproducibility analysis and how we conducted the reproducibility tests. We discuss our results in terms of rate of reproducibility, but more importantly also detail which issues we found

[2] It is important to reiterate that the founding editors of *CCR* expect that the online research compendium containing data, code, and results to be published on **GitHub or an equivalent service**. We interpret this as a general editorial expectation for data and code to be made available publicly.

[3] https://writings.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/

to make computational communication science (ir)reproducible. Surprisingly, at least to us, was the finding that missing data was the main culprit that makes studies in *CCR* fail our tests, **not any technological issue**. We end with reflections on what individual researchers and the (sub)field could do to improve reproducibility of research to ensure the credibility of results and improve the trust in academic work.

Transparency and Reproducibility

The first line of defense against irreproducibility, as stated in *Roadmap I*, is "an online appendix in a form that encourages reproducibility and reusability." At the very least, this step would lead to more transparency: Open Data and Open Materials (e.g. computer code). This call for data and code sharing can also be found in the recent calls for Open Science in communication science (Bowman & Spence, 2020; Dienlin et al., 2020; Lewis, 2019).

Transparency in scientific publications marks a significant advancement beyond the traditional opaque publication model—like a secret magic trick, research that completely lacks transparency is surely not reproducible. But a paper with all the code and data made available does not automatically become reproducible either (Peng, 2011). As defined by The Turing Way Community (2022) along with several other authors (i.e., Broman et al., 2017; Schoch et al., 2023), a result is reproducible "when the *same* analysis steps performed on the *same* dataset consistently produces the *same* answer." Therefore, one must perform the same analysis on the same dataset, i.e. execute the code with the data, and check whether the same answer can be obtained consistently. The difference between transparency and reproducibility is crucial as previous attempts to execute the code shared by researchers showed that most does not run without issues (Crüwell et al., 2023; Trisovic et al., 2022). Hence, despite great transparency, most of these studies are, in fact, not reproducible.

In order to attempt to reproduce findings and check results under different circumstances, the shared code must at least be executable by other parties using the same data—which hence also needs to be shared. An online appendix that can possibly enable executability was envisioned in *"Toward Open Computational Communication Science: A Practical Road Map for Reusable Data and Code"* (Van Atteveldt, Strycharz, et al., 2019, *Roadmap II* hereinafter) by a similar group of authors as *Roadmap I*. It is important to note that the authors of *Roadmap II* emphasize **reusability** of data and code, which they define as "allow[ing] and encourag[ing] other scholars to adapt them to their specific needs." *Roadmap II* does not focus on making data and code executable by others *per se*. However, certain ideas from it are helpful to achieve the same goal. For example, *Roadmap II* encourages the use of research compendia to share fully documented data and code, that the code and data should be version controlled and with unit tests, and that the computational environment should be preserved as a Dockerfile.

Reproducibility of Computational Communication Science

Half a decade has passed since *Roadmap I* defined the grand vision of the subfield computational communication science as the forerunner of transparent and reproducible research and *Roadmap II* laid out the practical steps towards this vision. Still, little is known about how successfully this grand vision was realized.

While we have some data on how (in)transparent the whole field of Communication is (Haim & Jungblut, 2023; Markowitz et al., 2021), there is no data on whether the subfield, as presented by the articles published in *CCR*, is improving the picture. This information can only be gleaned by executing the code shared by computational communication researchers—which we attempt in this study. Again, our most important goal is to qualitatively document all details that make computational communication science (ir)reproducible and to identify avenues to improve the reproducibility of the findings from the subfield.

Data and Approach

We attempt to reproduce all studies published in *CCR*. We set the date 2023-05-25 as the "snapshot date" of this study. The snapshot date means this current study is based on the published papers, materials (shared data and code), and technology available on this day. There are caveats to this claim (especially the last part) and we will explain these caveats in later sections. On this date or within the perimeter of a few months, the following actions were taken.

On the snapshot date, we preregistered the research question (*How many papers published in CCR are not computationally reproducible?*) [4] and protocol of the study.

In the protocol, we define the following events as reproducibility failure: (1) No shared code, (2) No shared data, (3) Code execution failure, despite

[4]Although our research question was preregistered as such (which we cannot change), it is our intention to study this question from the third-party perspective, i.e. *How many papers published in CCR are not computationally reproducible* ***by third parties, e.g. us?***

code rewrite, (4) Technically executable, but results with major deviations. These four criteria are an operationalization of computational reproducibility defined by Broman et al. (2017), Schoch et al. (2023), and The Turing Way Community (2022) (see above). Criterion 1 determines whether we can conduct the same analysis. Criterion 2 determines whether we can conduct the same analysis on the same data. Criterion 3 determines whether we can check if the analysis consistently produces the same answer. Criterion 4 determines whether the answer is indeed the same.

Our preregistered protocol established specific inclusion criteria for articles, limiting selection to those with results or outputs that can potentially be reproduced. Consequently, we only included articles that presented claims grounded in empirical analyses. Based on the preregistered protocol, a "postmortem guide" (see Appendix A) was authored as a guide to determine the computational reproducibility using the artifacts generated from a code execution attempt.

Data

All articles published in *CCR* up to the snapshot date were automatically scraped from the *CCR* website. In total, 47 articles were identified. These 47 articles were annotated by three coders for the following information:

- Type of article: Empirical analysis (substantive / methodological), Tool, Other (Theory paper etc.)
- Does the paper provide data on GitHub, OSF, or other repositories?
- Does the paper provide computer code on GitHub, OSF, or other repositories?

We identified 30 empirical papers (21 substantive papers and 9 methods papers), 13 tool presentations and 4 other papers. Among these 30 empirical articles, only 14 articles provide code and data. Therefore, up to this point we are not able to reproduce the findings from 46.7% of CCR papers, as 16 articles fulfilled Criterion 1 and/or 2 [1 only: 1, 2 only: 3, both: 12].

For the 14 papers initially coded with shared code and data, we will refer to them here as Articles A to N rather than their original titles, similar to Crüwell et al. (2023). This is because our focus is not these individual articles, but how the characteristics of data and code sharing practices impact the computational reproducibility. A complete list, however, is available in

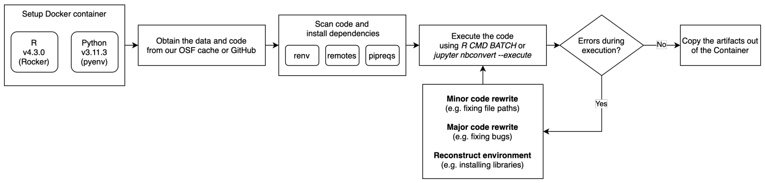
Setup Docker container
R v4.3.0 (Rocker)
Python v3.11.3 (pyenv)
Obtain the data and code from our OSF cache or GitHub
Scan code and install dependencies
renv
remotes
pipreqs
Execute the code using R CMD BATCH or jupyter nbconvert --execute
Errors during execution?
No
Copy the artifacts out of the Container
Yes
Minor code rewrite (e.g. fixing file paths)
Major code rewrite (e.g. fixing bugs)
Reconstruct environment (e.g. installing libraries)

Figure 1: Reproducibility Check Pipeline

the Online Appendix [5].

Although materials on OSF are versioned, it is not possible to obtain the versioned materials pragmatically. To solve this, on the snapshot date we archived all OSF materials of these 14 articles using the R package OSFR (Wolen et al., 2020). All reproducibility checks involving OSF materials were conducted with the archived copies.

Our Reproducibility Check Pipeline

We devise a reproducibility check pipeline for executing and documenting the reproduction of the included articles (see figure 1). This sections highlights our choices regarding the computational environment; how we resolved dependencies; how we executed the code; and how we checked the outcome.

The reproducibility check was conducted inside a Docker container based on ROCKER (Boettiger & Eddelbuettel, 2017), a container image based on Ubuntu Linux with preconfigured and versioned R. We added an optional layer of Python environment based on PYENV (pyenv authors, 2023). We pinned the version of R and Python to 4.3.0 and 3.11.3 respectively, the latest version as of the snapshot date. The goal of this dockerized pipeline is to fully automate the code execution in a standardized computational environment, that is broadly representative of what is used by computational communication researchers.

As most of the included articles do not provide a description of the computation environment, we automatically **resolved the dependencies** based on the shared code. The shared code was either from the archived OSF copies or the associated GitHub repository up to the latest commit before the snapshot date.

Shared R or Python code was scanned using RENV (Ushey, 2023) and PIPREQS (pipreqs authors, 2023) to identify the employed R and Python pack-

[5]Disclosure: One article in the set was authored by a coauthor of this paper. Because of that, the concerned coauthor was not involved in the reproducibility check of that article.

ages. For R packages, the system requirements (e.g. GNU SCIENTIFIC LIBRARY) were also queried using REMOTES (Csárdi et al., 2021).

Based on the scanned result, the software dependencies of the shared code were automatically installed inside the dockerized environment. We used Posit Public Package Manager [6] and pinned the date to the snapshot date. By doing so, we made sure the latest versions of the R and Python packages as of the snapshot date were installed.

The **code execution** part was developed as a single shell script file inside a Docker container that does the following: (1) Obtain the data and code from our OSF cache or GitHub; (2) Resolve and install dependencies automatically; (3) Optional: Code editing, using either GNU SED (Pizzini et al., 2018) or GNU PATCH (GNU patch authors, 2023)—to make the changes to the original material transparent and reproducible; (4) Execute the code; and (5) Copy the artifacts out of the Container for postmortem analysis (see Appendix A).

For the 14 articles, we attempted to **execute the shared code** in the above-mentioned Docker container. Exceptions are: 1) two Jupyter notebooks, which the authors of article B and article H recommend to run on Google Colaboratory; 2) Article J, for which the authors have prepared a description of their computational environment based on PACKRAT (Atkins et al., 2023). We followed the recommendations accordingly.

This procedure was iterative and the preregistered protocol allows the following three actions when the code execution attempt was failed

1. **Minor code rewrite**: When the execution failure was related to incorrect file paths, we edited the paths and attempted to rerun the code. We classified only the editing of file paths as minor code rewrite.
2. **Major code rewrite**: When the execution failure was related to code quality issues, i.e. bugs, we attempted to correct for the bugs and rerun. We classified this editing as major code rewrite.
3. **Reconstruct a customized computational environment**: When the execution failure was related to software libraries and our automatic pipeline did not resolve them, we attempted to create a customized computational environment for the code to run on.

If all three actions cannot make the code executable, the article satisfies Criterion 3 – *code execution failure, despite code rewrite.*

[6]https://posit.co/products/cloud/public-package-manager/

After code execution, we conducted a **postmortem analysis** based on the aforementioned postmortem guide (Appendix A): We compared the artifacts generated from the code execution attempt with either the archived artifacts from the original repositories or from the results on the papers to look for deviations. We follow Crüwell et al. (2023) to define minor deviations as "deviations in the decimals or obvious typographical errors." Deviations beyond decimals are classified as major deviations. We follow also Crüwell et al. (2023) to use the distinction between minor and major deviations as the cut-off point [7], i.e. papers with major deviations satisfy Criterion 4. The reasons for a paper satisfying Criterion 4 can either be the code itself cannot consistently produce results within the limit of deviations in decimals; or the code can produce consistent results but there are human errors in reporting.

Results

Quantitative Results

In this section, we present our findings in terms of how many of the attempts we consider a success or failure in reproducing the selected articles, followed by our qualitative assessment about both major and minor issues uncovered during our analysis in the next sections. Among the 14 articles we were able to evaluate, we confirmed that Articles H (with major code rewrite), E (with minor code rewrite) and M (with no code rewrite) are essentially reproducible. Articles B (with major code rewrite) and N (with minor code rewrite) are largely reproducible, except parts of the analyses reported in the Appendices, some data files were missing. Similarly, Article J is largely reproducible; except we detected a missing file in the feature extraction demo that stopped us from running it (the processed data is available). As these problems do not affect the analyses presented in the main body of the article, we deem Articles J, B and N partially reproducible [8]. Therefore, 20 % of articles in *CCR* that made claims based on empirical

[7]Crüwell et al. (2023) use three categories: "exactly reproducible" (no deviations), "essentially reproducible" (with minor deviations), and "largely not reproducible" (with major deviations). For simplicity, we merged "exactly reproducible" with "essentially reproducible" into one category. Although it did not state in the original definition, we understood that the minor deviations in the decimals should not affect the main conclusion. For example, rounding of p-value from 0.32 to 0.4 would be a minor deviation that does not affect the main conclusion. However, for precise measures, e.g. effect sizes, a change in from 0.32 to 0.4 might change the main conclusion.

[8]We did not preregister what to do with the code for analyses not in the main body of an article, i.e. supplementary analyses in online appendices. Our decision was to run the code

analysis and 42.9 % of the articles that shared code and data are at least partially reproducible.

Among all other articles, the code execution attempts were not successful despite code editing for Articles D, L, F, K, A, and I (satisfied Criterion 3). In case of Article C, we think that it might be possible to eventually reproduce it. Yet after rewriting individual files and creating a customized computational environment, as some required packages can only be installed with Python 3.8, we conceded when it became clear that we do not understand in which order to run the many different steps included in the sophisticated analysis pipeline. The code and data associated with Article G (with major rewrite) are executable but the output has major deviations (satisfied Criterion 4) [9].

Figure 2 shows an overview of the results within the set of all 47 published *CCR* articles. Below we summarize the major and minor issues that kept us from reproducing the tested articles.

Major Issues

Incomplete Sharing of Data and Code

This is the leading cause of irreproducibility among these 14 articles—which comes on top of the 16 that completely failed to share data and/or code. While these articles were initially categorized as having shared both data and code, upon attempting code execution, we found that certain elements of the code or data were, in fact, missing. These omissions seem like simple oversights, as the repositories appear to be comprehensive. It was only through our reproduction attempts that these small, yet consequential gaps came to light. The incompletely shared data and code render these articles satisfying Criterion 3 (code execution failure, despite code rewrite). As mentioned previously, even the three articles deemed partially reproducible base some of their analyses on data not included in the shared material.

The incomplete sharing manifests in different forms: (1) sharing only example data to demonstrate the feature extraction pipeline but no actual

beyond the main analysis anyway (if it is available). We made an *ad hoc* decision that irreproducibility in the supplementary analyses does not satisfy Criteria 3 and 4 but report it here. This can be considered a deviation from the preregistration. As Crüwell et al. (2023) consider main analyses only, our decision should be at least principled. However, given the field difference and supplementary analyses are common in our subfield, further studies should consider how to deal with (ir)reproducibility beyond the main analysis.

[9] We cannot confirm the actual reason for this. But it is very unlikely to be human errors in reporting because the inconsistent results are in the figures generated by code. Instead, we have reasons to believe that the code of Article G for data visualization is incomplete.

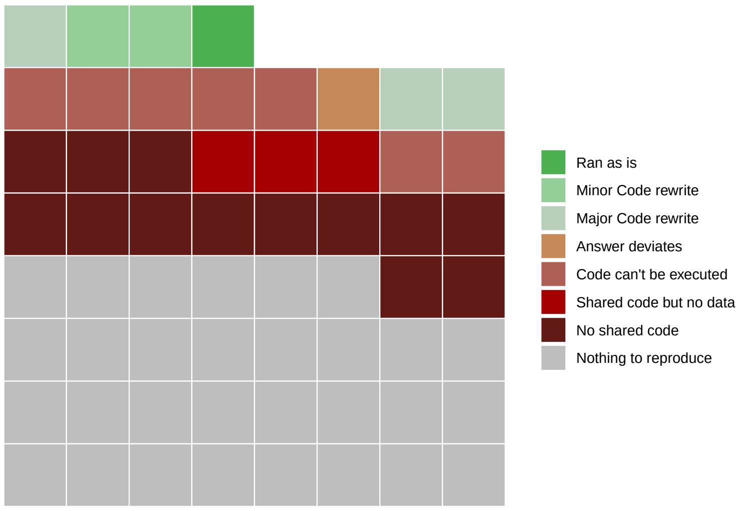

Figure 2: Types of (ir)reproducibility in *CCR* articles.

data as well as the code for data analysis (Articles L and F); (2) Some data is shared but that cannot be used to run the provided code (Article A); (3) Missing data columns in the provided data (Article I); (4) Data is complete but the code for generating some variables is missing (Article D); (5) Materials essential for running the code (e.g. list of stopwords) is not available (Article I).

Social Media Data Antics

A related issue to the incomplete sharing of data is the reliance on online access to social media data. When the access is no longer available, the inaccessibility manifests itself similarly to the issue above.

Article K provides over 50,000 Tweet IDs. Sharing IDs is the only permitted way of sharing data obtained from the Twitter API. In July 2023, we cannot rehydrate, i.e. retrieving the Tweets' complete information using their IDs, for free using the API provided by the X Corp. [10]. We can either

[10]https://web.archive.org/web/20230728143607/https://developer.twitter.com/en/

rehydrate 10,000 IDs per month for a total of USD600 (which would take 6 months); or rehydrate all IDs in one month for USD5000. Admittedly, we have neither the money nor the time.

A relatively minor issue related to this is that the feature extraction demos associated with Article B and Article H use YOUTUBE-DL to download videos from YouTube. This approach did not work in July 2023, with either the stable version of YOUTUBE-DL on PyPi or the development version on GitHub. Despite not being illegal, as some use cases, including academic research of publicly available videos, constitute fair use (Hennesy & Samberg, 2019), YouTube and other companies continue to fight YOUTUBE-DL with legal and technical means to hinder its usage. As already pointed out by Freelon (2018), availability of social media data is up to the whims of the companies who provide access to it and will probably further decrease in the future.

Minor Issues

Incomplete Sharing of Code

Even for the reproducible articles, some code for running minor parts in the analytical pipeline is missing. The missing parts concern with statistical tests (Article E), data visualization (Article M), and summary statistics (Article J).

Code Quality Issues

Our code execution attempts were in batch, e.g. using `R CMD BATCH` or `jupyter nbconvert --execute`. Some of the issue we encountered are likely because the code in question was developed **and checked** interactively only. This generates two issues. First, most visualization functions do not save the generated figures as files. Therefore, many researchers seem to rely on a tedious and error-prone manual process of saving the figures from the development environment such as RSTUDIO or JUPYTER NOTEBOOK. Second, scripts that should be run in sequence might demand unsaved objects from the previous script to be presence in the memory (or "workspace") for the current script to run (Article G). Another code quality issue we encountered and which required code editing, is the use of hard-coded absolute paths and other assumptions about the availability and structure of certain directories and files.

developer-terms/more-on-restricted-use-cases

Insufficient Documentation

Some code did not contain any documentation (e.g. Articles D and G). For those cases, we had to guess how to execute the code. Mostly, this was relatively simple, although, as explained above, Article C contains a rather sophisticated analysis pipeline, which we were not able to retrace, turning the lack of documentation from a minor into a major issue in that case.

Undocumented Computational Environment

Only three articles clearly documented the computational environment used for the original analysis. This lack of documentation of the computational environment generates two layers of problems. The first layer is the operating system. The code associated with Article H does not contain any information on the original computational environment. We were not able to execute the code in the dockerized Linux environment successfully, until we found out that the R function `list.files` behaves differently on Windows and Linux.

The second layer is the software library. The code associated with Article F, for example, does not provide any information on the computational environment, e.g. which version of PANDAS was used. In the code, many deprecated functions of PANDAS were used and we needed to rewrite the code substantially to make the code executable with the current version as of the snapshot date.

Discussion and Conclusion

This is so far the first study to evaluate the computational reproducibility of published communication research studies systematically. As is often the case in life, we derive good and bad news from our scrutiny of the reproducibility of articles published in *CCR*. We offer the bad news first: For *CCR*, a journal with a higher-than-average rate of data and code sharing, 80% of all empirical papers and 57.1 % of the articles that shared code and data are are not reproducible by third parties. However, our check is only possible because of the high data and code sharing rate (46.7%). Given the fact that the average of data and code sharing is less than 5% in other communication journals (Haim & Jungblut, 2023), the pool of studies included is certainly not representative and our number might be an **underestimation** of the actual number of the entire subfield of computational communication science. If we consider more communication journals, more papers would fall into Criterion 1 or 2 and a higher proportion of papers would be computationally irreproducible by third parties.

With 80% of the empirical papers being not reproducible by third parties (57.1 % were code and data was shared), the grand vision in *Roadmap I* has not (yet) been realized. On the contrary, the subfield is on the brink to descend into a reproducibility crisis like other Computational X disciplines, if not already in a full crisis. Corrective actions must be taken. Our study also gives flesh and blood to the recent calls for more Open Science for the field of communication science (Bowman & Spence, 2020; Dienlin et al., 2020; Lewis, 2019).

On that note, more bad news we derive from our check is a confirmation that simply providing code and data—what we might term 'basic transparency'—is just a starting point and not a guarantee of computational reproducibility. It is the meaningful first step to pass Criteria 1 and 2, but still not sufficient for fostering computational reproducibility: As Crüwell et al. (2023) and Trisovic et al. (2022) have noted before, most shared code is not executable without further actions. Our check confirmed that more than half of the shared code associated with *CCR* publications is not executable even with code rewrite. The most common is the incomplete sharing of data and code. Applying the software engineering term, this is a "runtime error", in contrast to "compile-time error" where we know before the code execution that the study is not reproducible, e.g. satisfying Criteria 1 and 2. "Runtime error" can only be found by the often tedious process of (manual) code execution. This manual check by third parties is not scalable: We (three researchers) took six months of our time just to check the computational reproducibility of 30 empirical papers. For comparison, ICA CM received 201 submissions in 2021 alone. Applying the same scale, we would take more than 3 years to check one year of ICA CM submissions manually.

The good news, on the other hand, is that there are still some published *CCR* papers that were found to be computationally reproducible. We can learn from these papers to improve the computational reproducibility of our works. See the "Recommendations for researchers" below.

Before diving into those recommendations, we would like to stress an important point from our analysis: **For most *CCR* articles, our code execution attempts were not blocked by technological issues**. Instead, it is due to the lack of reproducible materials in the first place (Criteria 1 and 2). The technical recommendations we give might take up most of the room in this part of our article, yet this is mostly a function of these tips being relatively straightforward to implement by junior researchers—such as ourselves. The field as a whole must not fall into the trap of so-called Technological Solutionism to "fix" the lack of transparency by perhaps yet another software tool to support computational reproducibility. The most effective way to save the subfield from the crisis is for the incentive system to change. We also provide several suggestions to the subfield to support greater research transparency, which we hope journal editors and senior researchers in the field will focus on.

Recommendations for Researchers

Execute and Check Code in Batch

The development of analytical pipeline with interpreted languages such as R and Python is usually interactive using tools such as the Read–Eval–Print Loop (REPL) or Jupyter Notebook. Interactive development is in fact an advantage of using interpreted languages and should be encouraged. However, we strongly recommend researchers **executing and checking** their developed code in batch (e.g. `R CMD Batch`, `Rscript`, `jupyter nbconvert --execute`, or `quarto render`). This is because interactive tools such as Jupyter Notebook do not enforce sequential execution and one can run the code in arbitrary order (Samuel & Mietchen, 2023). With batch execution, it is easier to identify simple errors such as missing files or undefined variables. Ideally, researchers design their project with the goal in mind that the entire analysis pipeline can be re-created by running scripts for data gathering, processing, analysis and summarising (e.g., in plots and tables) in order. Researchers can also use research compendium (next recommendation) and its associated features to ensure the sequential execution of code.

Use Research Compendium

Both *Roadmap I* and *Roadmap II* recommend the adoption of research compendium. The Turing Way (The Turing Way Community, 2022) defines a research compendium as "a collection of all digital parts of a research project including data, code, texts (protocols, reports, questionnaires, meta data). The collection is created in such a way that reproducing all results is straightforward."

In practice, a research compendium separates documentation, code, raw data, intermediate data, and results in a reasonable folder structure. Using a research compendium, together with tools such as HERE (Müller, 2020), or by adopting relative paths in RMarkdown/Quarto documents, can eliminate the most common reason for the code editing we did to make code run: incorrect file paths.

Two reproducible articles have their shared code and data organized as a research compendium: Articles H and N. Although not as a research compendium in a strict sense, Articles M, J, and B provide good documentation on how to reproduce the analysis and Article E separates code and data.

Several research compendium templates are already available, one of which developed by Communication researchers (Van Atteveldt et al., n.d.). Article N uses this template, which made the reproduction attempt considerable more straightforward. Inside the research compendium of Article N, the entire article was written in RMarkdown with PAPAJA (Aust & Barth, 2022) using the literate programming technique (mixing of prose with programming code, Knuth, 1984). Several of the reproducible articles (e.g. Articles B, E) also use some forms of literate programming. Literate programming eliminates the need for the error-prone copy and paste of figures and statistical tests and was recommended in *Roadmap II* and by Lewis (2019). Historically it was difficult for communication researchers to use literate programming, because communication journals usually accept Microsoft Word only (*CCR* once did). The recent introduction of Quarto and LaTeX templates by *CCR* is a boon for promoting literate programming to the subfield [11].

A good practice to ensure sequential execution of scripts is to name the scripts by their execution order, e.g. `01_setup.R`, `02_preprocess.R` [12], which Articles J and H use. Another approach is to use build management tools such as GNU MAKE (Stallman et al., 1988, Article N uses this) and DOIT (Schettino, 2021, which the original compendium uses).

Finally, the computational environment can also be documented inside the research compendium. It can be either as a document (e.g. Aricle N) or better, as a Dockerfile (e.g. Article J) to enable reproducible rebuild.

Reduce External Dependencies

We can dial back the clock for software version; but the same cannot be said about external dependencies. Schoch et al. (2023) define external dependencies as "parts in the research pipeline which some external stakeholders have complete control, but the researchers who create or use the research

[11] https://computationalcommunication.org/ccr/announcement/view/4. This article was written in LaTeX on Overleaf. We used SWEAVE (Leisch, 2002) together with KNITR (Xie, 2014) to enable literate programming

[12] See this presentation by Jenny Bryan: https://github.com/jennybc/how-to-name-files

pipeline does not." Because researchers—including the reproducibility checkers (e.g. us)—have no control over these external dependencies, almost nothing about these external dependencies can be done when they are the main culprits of irreproducibility. For instance, we cannot dial back the clock for Twitter API to its 2021 state for Article K (see Assenmacher et al., 2023); we cannot dial back the clock also for YouTube where YOUTUBE-DL still works for Articles B and H. It is only luck that the external dependencies in the analytical pipelines of some articles still work (Downloading a dictionary and language models during the code execution attempts for Articles L and F).

Although not in our cohort of articles, external dependencies also manifests as data analytic APIs, e.g. Google Translate, Botometer, and ChatGPT. The practice has been criticized because of the changing algorithms at the server end (Chan et al., 2020; Chen et al., 2023; Rauchfleisch & Kaiser, 2020). Like social media APIs, they can have the same destiny: Botometer, since June 2023, is no longer available due to the closure of the free Twitter API [13]. This serves as a wake-up call to the subfield that the reliance on external dependencies is a silent threat to computational reproducibility (Davidson et al., 2023; Schoch et al., 2023) and should be avoided at all cost. With the marveling of commercial LLM services such as ChatGPT in social science research, this threat becomes more imminent. Schoch et al. (2023) provide several alternatives on both the data and analytical fronts. In essence: (1) researchers should explore alternative ways of obtaining data to study online communication, e.g. data donation, and; (2) use open source software and pretrained models which can be locally deployed as often as possible.

Proactive Reproducibility

The retroactive approach to reproducibility is to make the analytical pipeline "reproducible" after the fact. This retroactive approach explains an often-cited reason for researchers' reluctance to share their code: time and effort to edit the code to make the code shareable (Cadwallader & Hrynaszkiewicz, 2022).

Other than the perceived effort for editing the code, we also observed during our execution attempts that the retroactive code editing can introduce new bugs into the shared code. One emblematic minor edit we did to correct for this kind of bugs was to change the call for the data file `ccr_data_share.csv` in the code back to `data.csv` for Article B, where only the file `data.csv` is available.

[13]https://botometer.osome.iu.edu/

We highly recommend that researchers should instead take a proactive approach to reproducibility, in which reproducibility is a built-in feature from the beginning. With this approach, code does not require any—or more realistically, any effortful—editing to be shareable. Data files are tiered by whether or not they can be shared to eliminate the risk of incomplete sharing or accidentally sharing of sensitive data. Automatic reproducibility checks such as continuous integration might also be used. The computational environment should also be captured by technologies such as Docker and Apptainer.

To take this approach, the analytical pipeline must be carefully designed and communication researchers might not have the knowledge to build it on their own. One approach is for communication researchers to collaborate with research software engineers when designing the analytical pipeline. Educational resources such as The Turing Way (The Turing Way Community, 2022) are also available.

Recommendations for the Subfield

доверяй, но проверяй

The Russian proverb доверяй, но проверяй (trust, but verify) was quoted by Ronald Reagan during the arms control negotiation with Mikhail Gorbachev in 1987. The same principle has been used by Willis and Stodden (2020) to evaluate the computational reproducibility of published scientific works. We **trust** the papers that we found to be irreproducible are probably computationally reproducible by the original authors on the machine they conducted the original analysis on. However, the goal should be that reproducibility can also be **verified** by anyone, everywhere, at any time. There is a need to move beyond the so-called "first-order computational reproducibility" (Schoch et al., 2023) or "repeatability" (McArthur, 2019), which cannot be independently verified.

Increasingly often, publication outlets require the reproducibility to be verified by third parties. Some publication outlets assign data editors to actively check for reproducibility of submissions (Vilhuber, 2023). In the realm of communication science, *Political Communication* is the first outlet to assign a data editor (Lawrence, 2022). This practice should be promoted.

As we mentioned previously, it took a lot of energy to check for the reproducibility of published items manually and we offer two solutions. The first solution is to make the data editor position funded to compensate for the time the data editor took to check for reproducibility, which *Political Communication* does. The second solution is to make the reproducibility check as effortless as possible. We provide several suggestions in the "Standardize the reproducible computational environment" section below.

Incentivize Data and Code Sharing

Non-sharing of code and data is still the most common reason for irreproducibility (53.3%). As many have argued (e.g. Rowhani-Farid et al., 2017; van Panhuis et al., 2014), this is largely not a technological issue. The data and code sharing infrastructures have been very mature in many countries, especially industrialized ones. A recent survey even found a counter-intuitive relationship that scientists' higher satisfactory with the resources for data sharing is associated with a decrease in willingness for them to share data (Borycz et al., 2023).

The willingness to share is an incentive issue (Akdeniz et al., 2023). Sharing of data and code is not required for publication in many communication journals (including *CCR*). Whether or not data and code sharing can impact citations in communication science remains inconclusive (Markowitz et al., 2021). There is *de facto* no incentive to share data and code. Schoch et al. (2023) argue that without significant incentives, sharing of data and code is mostly driven by moral responsibilities.

However, history told us that reliance on *bona fides* for behavioral changes does not scale up. The subfield must come up with carrots—hopefully not sticks—to induce researchers to share their data and code alongside their publications. We—the authors, some junior researchers—are not in the position to prescribe these field-level interventions for the subfield. However, communication researchers should believe in the possibility of the (sub)field to come up with a good solution to this issue, as evidenced by previous field-wise efforts such as the 70th Annual ICA Conference in promoting Open Science; and the data and software citation clause in ICA CM's Call for Papers since 2021.

We quoted доверяй, но проверяй previously. As a reply to Reagan, Gorbachev quoted "the reward of a thing well done is to have done it" (allegedly by Ralph Waldo Emerson). It also partially answers the incentive question.

Encourage Protected Access for Sensitive Data

Another obstacle for data sharing is the sharing of sensitive data. Most *CCR* articles we checked use sensitive data, such as social media data (obtained via restrictive APIs) and copy-righted data, and they are not publicly shareable. Communication researchers have come up with several solutions to this issue: Non-consumptive research (Gruber et al., 2023; Van Atteveldt et al., 2020) (providing access to data analysis capabilities of data without granting access to the often sensitive data itself) and sharing of encrypted data publicly (Van Atteveldt et al., n.d.) are two proposals. Practically in the cohort of *CCR* papers, researchers shared the processed intermediate data together with the code for data preprocessing, e.g. Articles J, E, and N.

A less technical solution is to promote "protected access" (or "controlled access"), a notion that is used by the journal *Psychological Science* [14]. Researchers can deposit sensitive data to approved protected access repositories (APARs) [15]. These repositories take care of the access control of deposited data. For example, if one would like to reproduce the analysis of a paper using a sensitive dataset deposited in an APAR, the APAR would require a formal application for data access. Therefore, the data desposited in APARs have a better care and the APARs have a paper-trail of who has a copy of the sensitive data. In other words, protected access is much more trustworthy than the so-called "available upon request" (Krawczyk & Reuben, 2012), both in terms of future access and the risk of leaking sensitive data.

As long as someone other than the original authors ("trusted third parties", e.g. data editor or other researchers) can have access to that sensitive data and reproduce the analysis, Schoch et al. (2023) classify this as second-order computational reproducibility.

Standardize the Reproducible Computational Environment

Our check shows that most of the shared code and data available can run in a standardized computational environment based on an off-the-shelf Docker image (Boettiger & Eddelbuettel, 2017), albeit most of the time code editing was needed. The computational environment we used can be rebuilt reproducibily and the edited code can run inside it automatically. Hence,

[14]https://www.psychologicalscience.org/publications/psychological_science/ps-submissions

[15]A list of APARs is available in https://osf.io/tvyxz/wiki/. Examples are Inter-university Consortium for Political and Social Research (ICPSR) at the University of Michigan, Research Data Repository at the University of Bristol, and Datorium at GESIS – Leibniz Institute for the Social Sciences.

our computational environment should fit a majority of the use cases. It points to a possibility of having a standardized, reproducible computational environment for running and checking the code of computational communication science.

To eliminate the need for manual code editing when checking, computational communication researchers would need to proactively develop the analytical pipeline for this standardized computational environment. If the subfield can even agree upon how the code should be executed inside this standardized computational environment (e.g. GNU MAKE), that would enable automatic code execution for checking the reproducibility of computational communication research, rather than the tedious manual code execution like we (and many other) did. This automatic process can be run anywhere Docker can run, e.g. Binder, GitHub Actions, and Heroku. The elimination of the tedious manual tasks can make the reproducibility verification of a submitted paper immediately available to the reviewers so that it can be taken into account.

Having a standardized computational environment requires a concerted effort, or else it would descent into the "XKCD 927" problem [16]. Although we demonstrate our proposed computational environment works practically, academic groups such as ICA CM together with and a consortium of journal editors are in a better position to derive this standard.

Coda: Trade-offs

"Programmers know the benefits of everything and the trade-offs of nothing", said Rich Hickey, the inventor of the Clojure language. We probably were too much in the *programmer mode* in the above discussion and therefore we quote Hickey here as a self-criticism. It is important to remind ourselves once again, computational reproducibility is not wholly a technological issue, but in large part a social issue. When dealing with social issues, we must know trade-offs more so than in dealing with technological issues.

The above discussion focuses only on the availability of data and code, as well as the software used for executing the code. An issue that was completely ignored in the above discussion is computational resources. For example, we took weeks to check the code of Article M on an off-the-shelf notebook computer due to the exceptionally long running time. For Article J, the demo code for video feature extraction demands specific hardware and drivers. We happened to get access to both, but nevertheless failed to

[16]XKCD 927, How Standards proliferate: https://xkcd.com/927/

get that code run due to a missing script in the shared code—at least we were unable to find it.

Our omission of discussing computational resources is intentional: we personally have inadequate resources to reproduce the most sophisticated approaches currently employed in the field—let alone running many of them in succession. But someone else might not face this issue. However, computational communication science will probably become more computational intensive in the future and it renders the task of computational reproducibility check like ours more demanding. Environmentally speaking, computational power should now be considered as a scare resource due to the greenhouse gas emission associated with it (Wu et al., 2022). Reproducibility checks like ours could be perceived as wasteful, if one sees no value in such duplicated efforts—the *re* in reproducibility.

Another omission in our discussion is that our suggestions increase the workload of computational communication researchers, whose time is already in scarcity. Getting reproducibility right is difficult, as seen in the code editing we needed to do even for articles we found to be reproducible. Although we have suggested automating this process as much as possible, there is still an upfront cost to learn about the tools and to write cleaner code—although we think these investments would produce considerable returns eventually. It circles back to the incentive question. Again, we have no solution.

Acknowledgements

We would like to thank *Computational Communication Science* for fostering transparent and reproducible research. We also would like to thank the original authors who submitted their works to *Computational Communication Science* and shared their code and data. Without their *bona fides*, this analysis of computational reproducibility is not possible.

References

Akdeniz, E., Borschewski, K. E., Breuer, J., & Voronin, Y. (2023). Sharing social media data: The role of past experiences, attitudes, norms, and perceived behavioral control. *Frontiers in Big Data, 5*. https://doi.org/10.3389/fdata.2022.971974

Assenmacher, D., Sen, I., Fröhling, L., & Wagner, C. (2023). *The end of the rehydration era - the problem of sharing harmful twitter research data*. ICWSM. https://doi.org/10.36190/2023.56

Atkins, A., Allen, T., Ushey, K., McPherson, J., Cheng, J., & Allaire, J. (2023). *Packrat: A dependency management system for projects and their r package dependencies* [R package version 0.9.1]. https://CRAN.R-project.org/package=packrat

Aust, F., & Barth, M. (2022). *papaja: Prepare reproducible APA journal articles with R Markdown* [R package version 0.1.1]. https://github.com/crsh/papaja

Bakker, B. N., Jaidka, K., Dörr, T., Fasching, N., & Lelkes, Y. (2021). Questionable and open research practices: Attitudes and perceptions among quantitative communication researchers. *Journal of Communication*, *71*(5), 715–738. https://doi.org/10.1093/joc/jqab031

Boettiger, C., & Eddelbuettel, D. (2017). An Introduction to Rocker: Docker Containers for R. *The R Journal*, *9*(2), 527. https://doi.org/10.32614/rj-2017-065

Borycz, J., Olendorf, R., Specht, A., Grant, B., Crowston, K., Tenopir, C., Allard, S., Rice, N. M., Hu, R., & Sandusky, R. J. (2023). Perceived benefits of open data are improving but scientists still lack resources, skills, and rewards. *Humanities and Social Sciences Communications*, *10*(1). https://doi.org/10.1057/s41599-023-01831-7

Bowman, N. D., & Spence, P. R. (2020). Challenges and best practices associated with sharing research materials and research data for communication scholars. *Communication Studies*, *71*(4), 708–716. https://doi.org/10.1080/10510974.2020.1799488

Broman, K., Cetinkaya-Rundel, M., Nussbaum, A., Paciorek, C., Peng, R., Turek, D., & Wickham, H. (2017). Recommendations to funding agencies for supporting reproducible research. *American statistical association*, *2*, 1–4.

Cadwallader, L., & Hrynaszkiewicz, I. (2022). A survey of researchers' code sharing and code reuse practices, and assessment of interactive notebook prototypes. *PeerJ*, *10*, e13933. https://doi.org/10.7717/peerj.13933

Chan, C.-h., Zeng, J., Wessler, H., Jungblut, M., Welbers, K., Bajjalieh, J. W., van Atteveldt, W., & Althaus, S. L. (2020). Reproducible extraction of cross-lingual topics (rectr). *Communication Methods and Measures*, 1–21. https://doi.org/10.1080/19312458.2020.1812555

Chen, L., Zaharia, M., & Zou, J. (2023). How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

Crüwell, S., Apthorp, D., Baker, B. J., Colling, L., Elson, M., Geiger, S. J., Lobentanzer, S., Monéger, J., Patterson, A., Schwarzkopf, D. S., Zaneva, M., & Brown, N. J. L. (2023). What's in a badge? a computational reproducibility investigation of the open data badge policy in one issue of psychological science. *Psychological Science*, 095679762211408. https://doi.org/10.1177/09567976221140828

Csárdi, G., Hester, J., Wickham, H., Chang, W., Morgan, M., & Tenenbaum, D. (2021). *Remotes: R package installation from remote repositories, including 'github'* [R package version 2.4.2]. https://CRAN.R-project.org/package=remotes

Davidson, B. I., Wischerath, D., Racek, D., Parry, D. A., Godwin, E., Hinds, J., van der Linden, D., Roscoe, J. F., & Ayravainen, L. (2023). Social media APIs: A quiet threat to the advancement of science.

Dienlin, T., Johannes, N., Bowman, N. D., Masur, P. K., Engesser, S., Kümpel, A. S., Lukito, J., Bier, L. M., Zhang, R., Johnson, B. K., Huskey, R., Schneider, F. M., Breuer, J., Parry, D. A., Vermeulen, I., Fisher, J. T., Banks, J., Weber, R., Ellis, D. A.,

… de Vreese, C. (2020). An agenda for open science in communication. *Journal of Communication*, *71*(1), 1–26. https://doi.org/10.1093/joc/jqz052

Freelon, D. (2018). Computational research in the post-api age. *Political Communication*, *35*(4), 665–668. https://doi.org/10.1080/10584609.2018.1477506

GNU patch authors. (2023). GNU patch. https://savannah.gnu.org/projects/patch/

Gruber, J. B., Van Atteveldt, W., & Welbers, K. (2023). *Sharing is caring (about research): Three avenues for sharing (copyrighted) text collections and the need for non-consumptive research*. https://github.com/JBGruber/paper_nonconsumptive/blob/opted/OPTED-D7.6.pdf

Haim, M., & Jungblut, M. (2023). How open is communication science? open-science principles in the field. *Annals of the International Communication Association*, 1–20. https://doi.org/10.1080/23808985.2023.2201601

Hennesy, C., & Samberg, R. (2019). Law and literacy in non-consumptive text mining: Guiding researchers through the landscape of computational text analysis. *Copyright Conversations: Rights Literacy in a Digital World*. https://escholarship.org/uc/item/55j0h74g

Hilbert, M., Barnett, G., Blumenstock, J., Contractor, N., Diesner, J., Frey, S., Gonzalez-Bailon, S., Lamberso, P., Pan, J., Peng, T.-Q., Shen, C., Smaldino, P. E., Van Atteveldt, W., Waldherr, A., Zhang, J., & Zhu, J. J. H. (2019). Computational communication science: A methodological catalyzer for a maturing discipline. *International Journal of Communication*. https://ijoc.org/index.php/ijoc/article/view/10675/2764

Hothorn, T., & Leisch, F. (2011). Case studies in reproducibility. *Briefings in Bioinformatics*, *12*(3), 288–300. https://doi.org/10.1093/bib/bbq084

Hutson, M. (2018). Missing data hinder replication of artificial intelligence studies. *Science*. https://doi.org/10.1126/science.aat3298

Ioannidis, J. P. A., Allison, D. B., Ball, C. A., Coulibaly, I., Cui, X., Culhane, A. C., Falchi, M., Furlanello, C., Game, L., Jurman, G., Mangion, J., Mehta, T., Nitzberg, M., Page, G. P., Petretto, E., & van Noort, V. (2009). Repeatability of published microarray gene expression analyses. *Nature Genetics*, *41*(2), 149–155. https://doi.org/10.1038/ng.295

Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Krawczyk, M., & Reuben, E. (2012). (Un)Available upon Request: Field Experiment on Researchers' Willingness to Share Supplementary Materials. *Accountability in Research*, *19*(3), 175–186. https://doi.org/10.1080/08989621.2012.678688

Lawrence, R. G. (2022). Editor's note. *Political Communication*, *40*(1), 1–3. https://doi.org/10.1080/10584609.2022.2155758

Leisch, F. (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. *COMPSTAT: Proceedings in computational statistics*, 575–580.

Lewis, N. A. (2019). Open communication science: A primer on why and some recommendations for how. *Communication Methods and Measures*, *14*(2), 71–82. https://doi.org/10.1080/19312458.2019.1685660

Markowitz, D. M., Song, H., & Taylor, S. H. (2021). Tracing the adoption and effects of open science in communication research*. *Journal of Communication*. https://doi.org/10.1093/joc/jqab030

Matthes, J., Marquart, F., Naderer, B., Arendt, F., Schmuck, D., & Adam, K. (2015). Questionable research practices in experimental communication research: A systematic analysis from 1980 to 2013. *Communication Methods and Measures*, *9*(4), 193–207. https://doi.org/10.1080/19312458.2015.1096334

McArthur, S. L. (2019). Repeatability, reproducibility, and replicability: Tackling the 3R challenge in biointerface science and engineering. *Biointerphases*, *14*(2). https://doi.org/10.1116/1.5093621

Mede, N. G., Schäfer, M. S., Ziegler, R., & Weißkopf, M. (2020). The "replication crisis" in the public eye: Germans' awareness and perceptions of the (ir)reproducibility of scientific research. *Public Understanding of Science*, *30*(1), 91–102. https://doi.org/10.1177/0963662520954370

Müller, K. (2020). *Here: A simpler way to find your files* [R package version 1.0.1]. https://CRAN.R-project.org/package=here

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226–1227. https://doi.org/10.1126/science.1213847

pipreqs authors. (2023). Pipreqs: Generate pip requirements.txt file based on imports of any project. https://github.com/bndr/pipreqs

Pizzini, K., Bonzini, P., Meyering, J., & Gordon, A. (2018). GNU sed, a stream editor. https://www.gnu.org/software/sed/manual/sed.pdf

pyenv authors. (2023). Pyenv: Simple python version management. https://github.com/pyenv/pyenv

Rauchfleisch, A., & Kaiser, J. (2020). The false positive problem of automatic bot detection in social science research. *PLOS ONE*, *15*(10), e0241045. https://doi.org/10.1371/journal.pone.0241045

Rowhani-Farid, A., Allen, M., & Barnett, A. G. (2017). What incentives increase data sharing in health and medical research? a systematic review. *Research Integrity and Peer Review*, *2*(1). https://doi.org/10.1186/s41073-017-0028-9

Samuel, S., & Mietchen, D. (2023). Computational reproducibility of jupyter notebooks from biomedical publications. *arXiv preprint arXiv:2308.07333*.

Schettino, E. N. (2021). pydoit/doit: Task management and automation tool (python). https://doi.org/10.5281/ZENODO.4892135

Schoch, D., Chan, C.-h., Wagner, C., & Bleier, A. (2023). Computational reproducibility in computational social science. https://doi.org/10.48550/ARXIV.2307.01918

Stallman, R. M., McGrath, R., & Smith, P. (1988). GNU make.

The Turing Way Community. (2022). The Turing Way: A handbook for reproducible, ethical and collaborative research. https://doi.org/10.5281/zenodo.3233853

Trisovic, A., Lau, M. K., Pasquier, T., & Crosas, M. (2022). A large-scale study on research code quality and execution. *Scientific Data*, *9*(1). https://doi.org/10.1038/s41597-022-01143-6

Ushey, K. (2023). *Renv: Project environments* [R package version 0.17.3]. https://CRAN.R-project.org/package=renv

Van Atteveldt, W., Althaus, S., & Wessler, H. (2020). The trouble with sharing your privates: Pursuing ethical open science and collaborative research across national jurisdictions using sensitive data. *Political Communication*, *38*(1-2), 192–198. https://doi.org/10.1080/10584609.2020.1744780

Van Atteveldt, W., Loecherbach, F., Steijaert, M., van der Velden, M. A., Welbers, K., Kroon, A. C., Strycharz, J., & Trilling, D. (n.d.). CCS Compendium: A template and tool support for open and transparent science. https://i.amcat.nl/compendium.pdf

Van Atteveldt, W., Margolin, D., Shen, C., Trilling, D., & Weber, R. (2019). A roadmap for Computational Communication Research [Editorial to inaugural issue]. *Computational Communication Research*, *1*(1), 1–11. https://doi.org/10.5117/CCR2019.1.001.VANA

Van Atteveldt, W., Strycharz, J., Trilling, D., & Welbers, K. (2019). Toward open computational communication science: A practical road map for reusable data and code. *International Journal of Communication*, *13*, 20.

van Panhuis, W. G., Paul, P., Emerson, C., Grefenstette, J., Wilder, R., Herbst, A. J., Heymann, D., & Burke, D. S. (2014). A systematic review of barriers to data sharing in public health. *BMC Public Health*, *14*(1). https://doi.org/10.1186/1471-2458-14-1144

Vilhuber, L. (2023). Reproducibility and transparency versus privacy and confidentiality: Reflections from a data editor. *Journal of Econometrics*.

Willis, C., & Stodden, V. (2020). Trust but verify: How to leverage policies, workflows, and infrastructure to ensure computational reproducibility in publication. *Harvard Data Science Review*, *2*(4). https://doi.org/10.1162/99608f92.25982dcf

Wolen, A. R., Hartgerink, C. H., Hafen, R., Richards, B. G., Soderberg, C. K., & York, T. P. (2020). osfr: An R interface to the open science framework. *Journal of Open Source Software*, *5*(46), 2071. https://doi.org/10.21105/joss.02071

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Aga, F., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H.-H., … Hazelwood, K. (2022). Sustainable AI: Environmental implications, challenges and opportunities. In D. Marculescu, Y. Chi, & C. Wu (Eds.), *Proceedings of machine learning and systems* (pp. 795–813). https://proceedings.mlsys.org/paper_files/paper/2022/file/462211f67c7d858f663355eff93b745e-Paper.pdf

Xie, Y. (2014). Knitr: A comprehensive tool for reproducible research in R [ISBN 978-1466561595]. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Chapman; Hall/CRC.

Zeng, J., & Chan, C.-h. (2023). Envisioning a more inclusive future for Digital Journalism: A diversity audit of journalism studies (2013–2021). *Digital Journalism*, *11*(4), 609–629. https://doi.org/10.1080/21670811.2023.2182803

Appendix A Postmortem Guide

Postmortem guide

Step 1: No Code and/or No Data -> **[rating: Not verifiable], STOP**; otherwise **goto** Step 2

Step 2: Run the code

If executable -> **[executable: 1]** -> **goto** Step 3; otherwise **goto** step 2.1

Step 2.1: Edit the paths and rerun

If executable -> **[executable: 1], [rewrite / edit: 1] [document all edited lines]** -> **goto** Step 3; otherwise **goto** Step 2.2

Step 2.2: Edit the code and rerun

If executable -> **[executable: 1], [rewrite / edit: 2] [document all edited lines]** -> **goto** Step 3

If not executable ->

if the non-execution related to code issues -> **[rewrite / edit: 2], [executable: 0]** -> **goto** Step 5

If the non-execution related to external libraries -> **goto** Step 2.3

Step 2.3: Produce a customized Docker image and rerun

If executable -> **[executable: 1], [rewrite / edit: depends on step 2.2] [dockerize: 1]** -> **goto** Step 3; otherwise **[dockerize: 1], [executable 0]** -> **goto** Step 5

Step 3: Check for code completeness with the paper

If some parts are missing -> **[code_type: insufficient]** -> **goto** Step 4; otherwise **[code_type: complete]** -> **goto** Step 4

Step 4: Check for deviations with the paper

Unknown, uncheckable -> **[major deviations: 99]** -> **goto** step 5

Any result with any minor deviation ("minor deviations in the decimals or obvious typographical errors") -> **[major deviations: 2]** -> **goto** Step 5

Any result with any major deviation (any deviation larger than decimals) -> **[major deviations: 1]** -> **goto** step 5

Otherwise, i.e. exact -> **[major deviations: 0]** -> **goto** Step 5

Step 5: Code the following variables: *laptop, special equipment, external_dependencies, doc_steps, doc_literate, doc_literate_complete*; give recommendations, log name, **STOP**