Replication materials are great but software versioning still poses a problem for open science

Taylor J. Wright* Bruno Rodrigues †

June 14, 2023

*Brock University, twright3@brocku.ca
†Ministry of Higher Education and Research, Luxembourg

1 Introduction

In recent years, the ''replication crisis" in the social sciences has become mainstream, even featured in the popular press. Concerns over how well studies hold up to replication in other settings has lead to large scale initiatives to document how trustworthy the existing stock of evidence is (cite Camerer and Nosek). While the replicability of studies in important, there has been a tandem movement discussing how reproducible research results are. There are various definitions, but for broadly speaking replication is in some sense re-testing a hypothesis while changing an element of previous research (e.g. the sample, or the estimating equation) whereas reproducibility is following a study's protocol exactly and obtaining the results presented in the study (insert a citation with some definitions here, perhaps Lars' blog or Michael Clemens JES paper?). In our view, reproducibility is an insufficient but necessary condition for replication—that is, it does not make sense to spend resources on replciation if research is not reproducible.

These concerns about the reproducibility and replicability of social science research have prompted a push for journals to require the publication of research materials that accompany academic research (see, for example, the 2014 Joint Editors Transparency Statement which was signed by editors of 27 leading political science journals: https://www.dartstatement.org/2014-journal-editors-statement-jets). Specifically, the provision of underlying data and scripts used for data preparation and analysis. However, even if these materials are provided (and even when in place these policies do not have perfect compliance (Philip (2010), Stockemer, Koehler, and Lentz (2018))), the regular software updates and new version releases can result in the replication materials failing to faithfully reproduce the authors' results or even run at all (Simonsohn (2021) presents several examples of R changes that could break scripts).

In this paper we ppresent a case study of an article published in Journal of Politics in January 2022 titled, "Multiracial Identity and Political Preferences", Davenport, Franco, and Iyengar (2022), that details that replication challenges arising from changes in the statistical software R. We were unable to reproduce the authors' results using either the current version of R, or the version that the authors indicate they used. The lack of reproducibility arose due to a change in the defaults used by base R when generating random numbers starting in version 3.6.0.

We contribute to the existing literature…

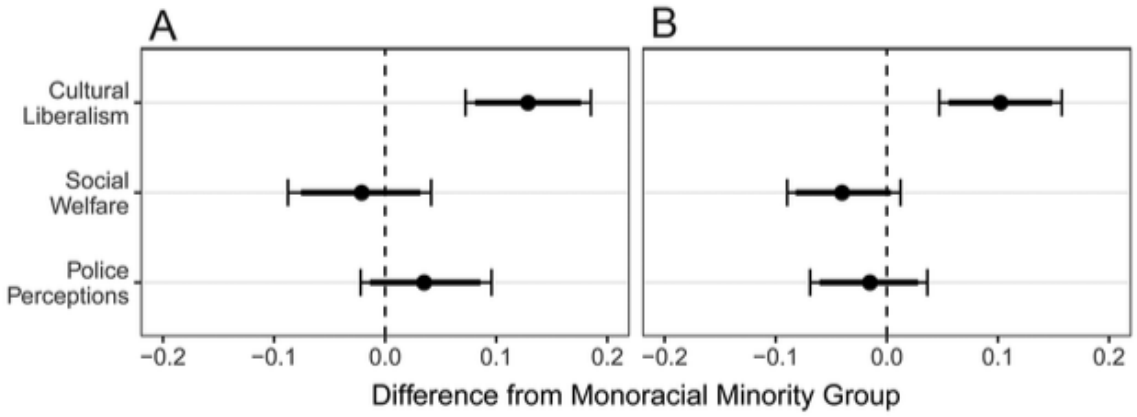
The rest of the article proceeds as follows: Section 2 walks through the reproducibility issues in Davenport, Franco, and Iyengar (2022); Section 3 discusses currently available tools and best practices (e.g. Docker and R packages such as `renv`, `groundhog`) for ensuring that replication materials continue to faithfully reproduce research results, despite post-publication changes in the tools used; and Section 4 concludes.

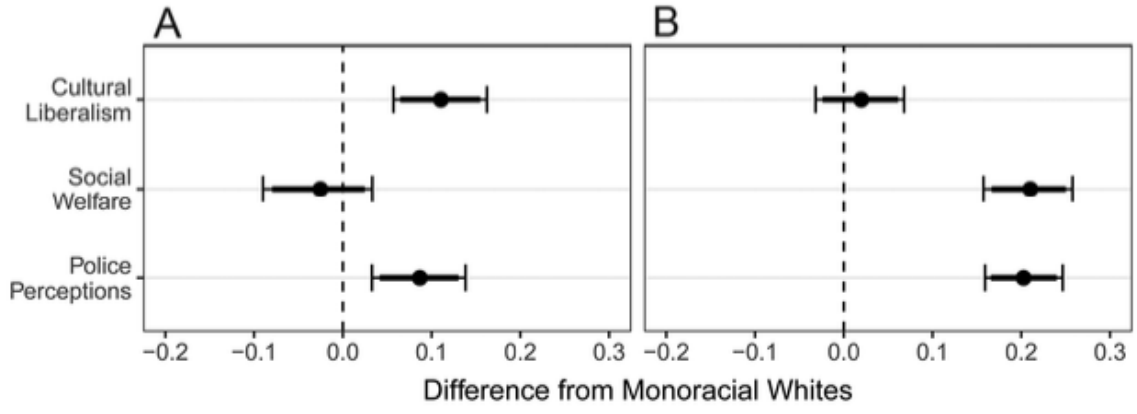
2 Reproduction

2.1 Illustrating the issue with software versioning

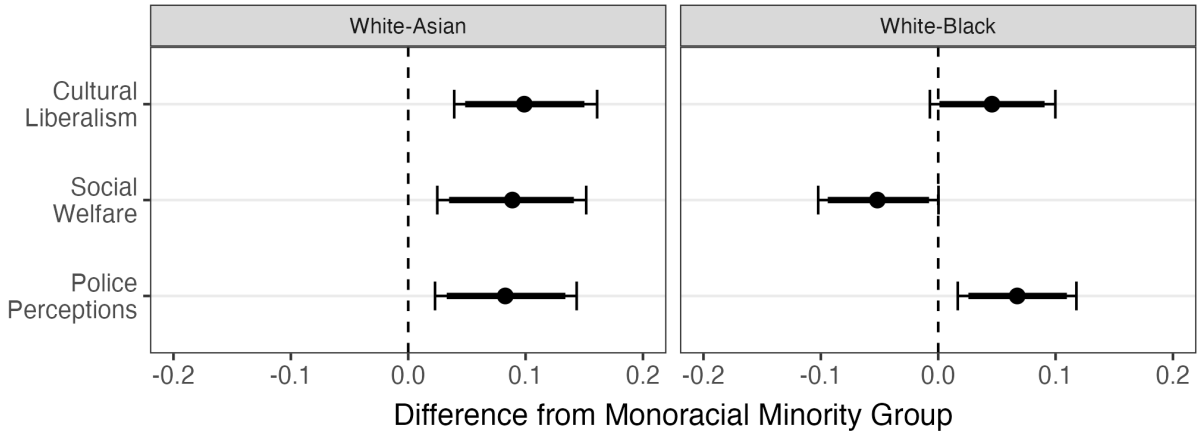
A key thing here is that I don't think we want to be too hostile sounding towards these authors, they had readme files and reproducibility materials available. It's just that manual entry human-error hobgoblins got them with the R versioning.

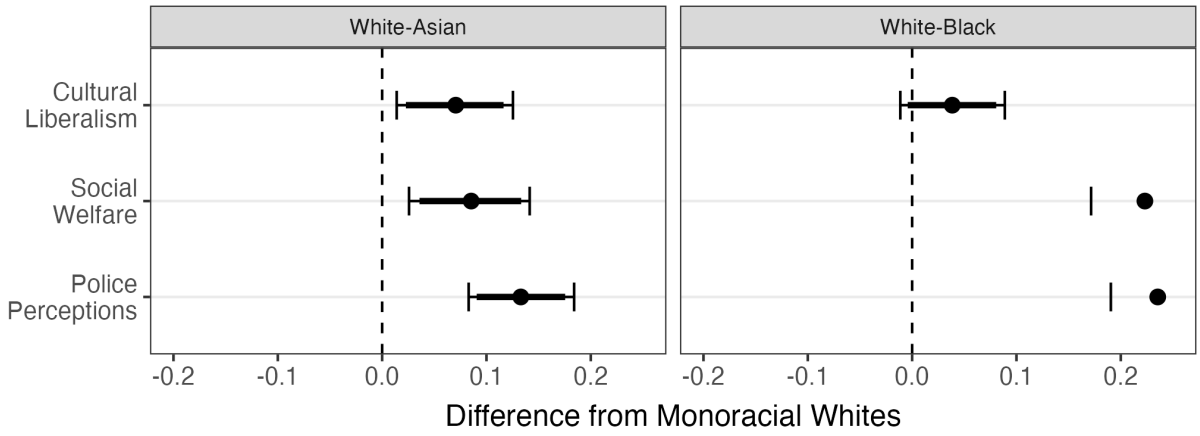
- Brief discussion of authors' paper and context
- The results of their code using stated software version in documentation (just screenshot right now)



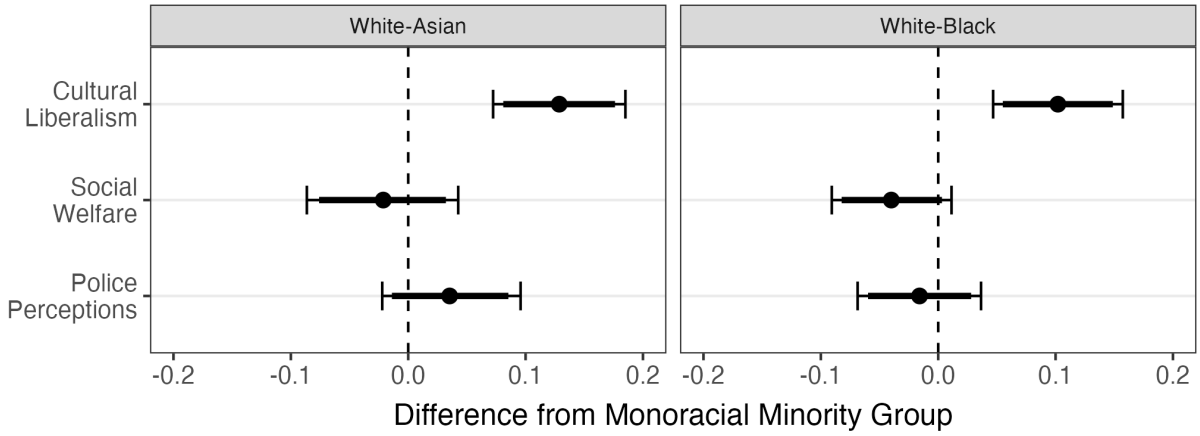


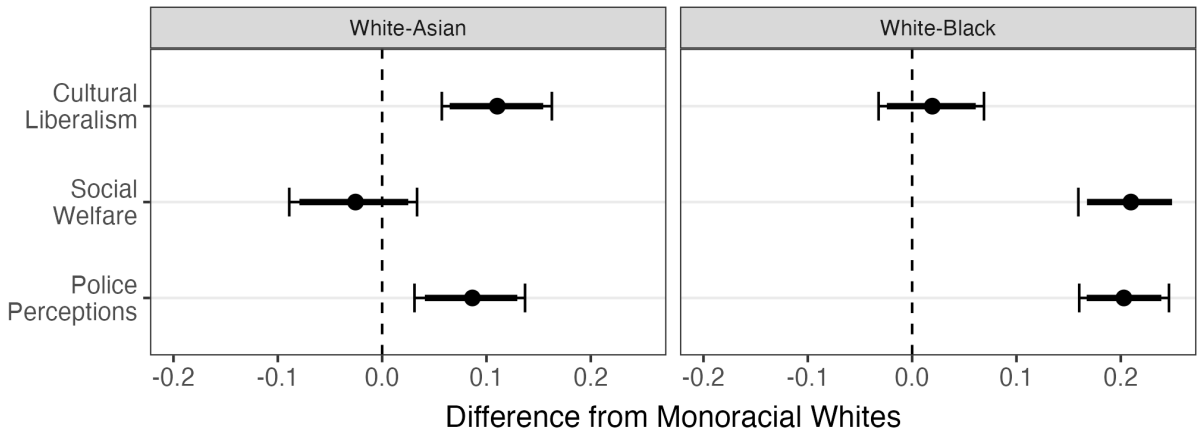
- The results of their code using later software version (post 3.6.0)





- Results of their code using earlier software version (pre 3.6.0)





The authors note that they use R version 3.6.2. Issue seems to be the weights the authors use are non-integer and Zelig uses sample() in that case which following changes to base R yields different results in 3.6.x and 3.5.x (see http://docs.zeligproject.org/articles/weights.html). Prior to R 3.6.x RNGkind(sample.kind = "Rounding") was the default behaviour but after 3.6.0 the sample function's new default behaviour is RNGkind(sample.kind = "Rejection") (see https://blog.revolutionanalytics.com/2019/05/whats-new-in-r-360.html).

Soemthing about how there are ways to ensure that the estimates are consistent (changing the weights to integers or setting the RNGkind to be backwards compatible) but documenting the correct version of R used in the analysis is probably the easiest way. Segue into…

3 Discussion

3.1 The problem is the seed?

The table below shows the quantiles of the means obtained from 100 runs with 100 different random seeds on R version 3.5 for the `m2m` models. We should check where the coefficients from the original paper fall in that distribution, and see if coefficients should vary so much simply from changing the seed?

race	model	q_005_mean	q_20_mean	q_40_mean	q_50_mean	q_60_mean	q_80_mean	q_95_mean
White-Asian	Police Perceptions	-0.01526542	-0.001102816	0.01759581	0.02586489	0.03099894	0.04413257	0.065472091
White-Asian	Social Welfare	-0.02072013	-0.006649473	0.01944720	0.03262676	0.03905241	0.05812758	0.079513026
White-Asian	Cultural Liberalism	0.07802019	0.098207422	0.11239090	0.12270775	0.13180765	0.14661806	0.169542042
White-Black	Police Perceptions	-0.02331918	-0.001568411	0.01533537	0.02181459	0.02680799	0.04384726	0.067172559
White-Black	Social Welfare	-0.08351311	-0.062048288	0.04489936	0.04136138	0.03611174	0.02119538	0.001704422
White-Black	Cultural Liberalism	0.03562363	0.060164495	0.07291920	0.07738611	0.08234863	0.10201330	0.119382119

The table below shows the quantiles of the means obtained from 100 runs with 100 different random seeds on R version 3.5 for the `m2w` models:

race	model	q_005_mean	q_20_mean	q_40_mean	q_50_mean	q_60_mean	q_80_mean	q_95_mean
White-Asian	Police Perceptions	0.046735795	0.067470058	0.08117265	0.09007913	0.09895873	0.11565896	0.13773174
White-Asian	Social Welfare	-0.004927122	0.020943227	0.03311780	0.05020756	0.05851728	0.07343161	0.10264449
White-Asian	Cultural Liberalism	0.046796515	0.062335404	0.08756450	0.09205009	0.09816402	0.11266966	0.13290792
White-Black	Police Perceptions	0.169510723	0.189838189	0.20124209	0.20983966	0.21606705	0.23216549	0.26767669
White-Black	Social Welfare	0.154936922	0.177350470	0.19262474	0.19770454	0.20252183	0.22472549	0.24515122
White-Black	Cultural Liberalism	-0.009211098	0.008618765	0.02023309	0.02625295	0.02985881	0.04424369	0.06153429

3.2 Rebuilding the original development environment using Docker

Replicating results from past studies is quite challenging, for many reasons. This articles focuses on one of these reasons: results cannot be reproduced because of changes introduced in more recent versions of the software used for analysis, despite the availability of both data and replication scripts.

To replicate the results from the original study and to pinpoint the impact of the change introduced in R 3.6.0, we chose to use Docker. Docker is a containerization tool which enables one to build so-called *images*. These images contain a software product alongside its dependencies and even pieces of a Linux operating system. To use the software product, customers in turn only need to be able to run Docker *containers* from the image definition. Containerization tools such as Docker solved the "works on my machine" problem: this problem arises when software that works well on the development machine of the developer fails to run successfully

on a customer's machine. This usually happens because the customer's computer does not have the necessary dependencies to run the software (which are traditionally not shipped alongside the software product).

A research project can also be seen as a *software product*, and suffers thus from the same "works on my machine" problem as any other type of software. Containerization tools offer a great opportunity for reproducibility in research: instead of just sharing a replication script, authors can now easily share the right version of the software used to produce these scripts, as well as the right version of the used libraries by building and sharing a Docker image (or at least provide the necessary blueprint to enable others to do so, as we will discuss below). Future researchers looking to replicate the results can now simple run a container from the provided image (or build an image themselves if the original authors provided the required blueprints).

Concretely, to build a Docker image, a researcher writes a so-called *Dockerfile*. Here is an example of a very simple Dockerfile:

```
FROM rocker/r-ver:4.3.0

CMD ["R"]
```

This Dockerfile contains two lines: the first line states which Docker image we are going to use as a base. Our image will be based on the 'rocker/r-ver:4.3.0' image. The Rocker project is a repository containing many images that ship different versions of R and packages pre-installed: so the image called `r-ver:4.3.0` is an image that ships R version 4.3.0. The last line states which command should run when the user runs a container defined from our image, so in this case, simply the R interactive prompt. Below is an example of a Dockerfile that runs an analysis script:

```
FROM rocker/r-ver:4.3.0

RUN R -e "install.packages('dplyr')"

RUN mkdir /home/research_project

```

```
RUN mkdir /home/research_project/project_output

RUN mkdir /home/research_project/shared_folder

COPY analyse_data.R /home/research_project/analyse_data.R

RUN cd /home/research_project && R -e "source('analyse_data.R')"

CMD mv /home/research_project/project_output/* /home/research_project/shared_folder/
```

This Dockerfile starts off from the same base image, an image that ships R version 4.3.0, than it install the `{dplyr}` package, a popular R package for data manipulation and it creates three directories:

- `/home/research_project`
- `/home/research_project/project_output`
- `/home/research_project/shared_folder`.

Then, it copies the `analyse_data.R` script, which contains the actual analysis made for the purposes of the research project, into the Docker image. The second-to-last line runs the script, and the last line moves the outputs generated from running the `analyse_data.R` script to a folder called `shared_folder`. It is important to say that `RUN` statements will be executed as the image gets built, and `CMD` statements will be executed as a container runs. Using this Dockerfile, an image called `research_project` can be built with the following command:

```
docker build -t research_project .
```

This image can then be archived and shared for replication purposes. Future researchers can then run a container from that image using a command such as:

```
docker run -d -it --rm --name research_project_container \
```

```
  -v /host/machine/shared_folder:/home/research_project/shared_folder:rw \
  research_project
```

The container, called `research_project_container` will execute the `CMD` statement from the Dockerfile, in other words, move the outputs to the `shared_folder`. This folder is like a door between the machine that runs the container and the container: by doing this, the outputs generated within the container can now be accessed from the host's computer.

The image built by the process above is immutable: so as long as users can run it, the outputs produced will be exactly the same as when the original author ran the original analysis. However, if the image gets lost, and needs to be rebuilt, the above Dockerfile will not generate the same image. This is because the Dockerfile, as it is written above, will download the version of `{dplyr}` that is current at the time it gets built. So if a user instead builds the image in 5 years, the version of `{dplyr}` that will get downloaded. The version of R, however, will forever remain at version 4.3.0.

So to ensure that future researchers will download the right versions of packages that were originally used for the project, the original researcher also needs to provide a list of packages and their versions. This can be quite tedious if done by hand, but thankfully, there are ways to generate such lists very easily. The `{renv}` package for the R programming language provides such a function. Once the project is done, one simply needs to call:

```
renv::init()
renv::hydrate()
renv::snapshot()
```

to generate a file called `renv.lock`. This file contains the R version that was used to generate it, the list of packages that were used for the project, as well as their versions and links to download them. This file can be used to easily install all the required packages in the future by simply running:

```
renv::restore()
```

A researcher can thus add the following steps in the Dockerfile to download the right packages when building the image:

```
COPY renv.lock /home/research_project/renv.lock

RUN R -e "setwd('/home/research_project);renv::init();renv::restore()"
```

However, what should researchers that want to replicate a past study do if the original researcher did not provide a Dockerfile nor an `renv.lock` file? This is exactly the challenge that we were facing when trying to replicate the results of Davenport, Franco, and Iyengar (2022). We needed to find a way to first, install the right version of R, then the right version of the packages that they used, run their original script, and then repeat this procedure but this time on a recent version of R.

3.3 Reproducibility isn't just version and package management — using {targets} to improve readibility and reproducibility

4 Conclusion

References

Davenport, Lauren, Annie Franco, and Shanto Iyengar. 2022. "Multiracial identity and political preferences." *The Journal of Politics* 84 (1): 620–24.

Philip, Glandon. 2010. "Report on the American Economic Review Data Availability Compliance Project." *Unpublished Manuscript*.

Simonsohn, Uri. 2021. "[95] Groundhog: Addressing The Threat That R Poses To Reproducible Research." *Data Colada*. http://datacolada.org/95.

Stockemer, Daniel, Sebastian Koehler, and Tobias Lentz. 2018. "Data access, transparency, and replication: New insights from the political behavior literature." *PS: Political Science & Politics* 51 (4): 799–803.