Computational Reproducibility

Jürgen Schneider

2025-01-22

Table of contents

General

Here you'll find information on how to make your data analyses computational reproducibile.

- Basic requirement: **Share** data and analyses (see `closed_data()` function from WORCS or synthpop package in case you can't share data)
- Set up your work as a **'project'**, where all related files (e.g., data, scripts, results) are stored together in a single folder (as with R-projects) or in a single file (as with JASP and jamovi). Avoid working with isolated files, and use **relative paths** to connect files within the project.
- Use a clear folder structure and readme files

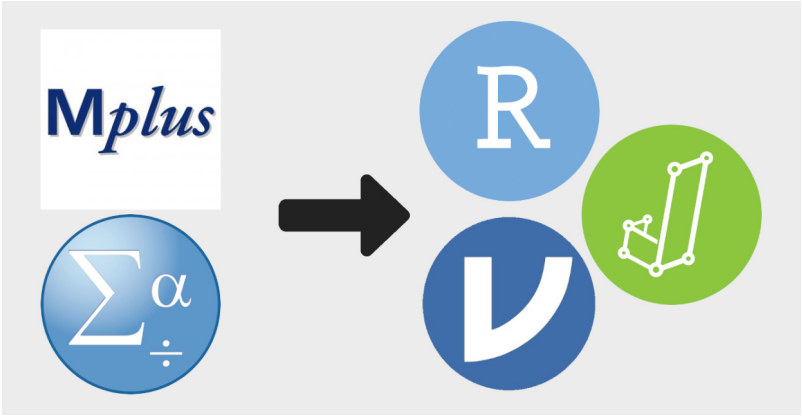
The Workflow for Open Reproducible Code in Science (WORCS) is an excellent framework that also integrates recommendations we give here.

Contents

See also navigation menu on the left.

Open file formats and softwareEnsuring cost-free access to your analyses Input-output-documentsMaking analyses comprehensible for yourself and others Containerization and version managementMaking analyses system-independent and sustainable

1 Open file formats and software



Reproducibility in research relies on sharing both the steps used to analyze data and the data itself. Using open file formats and free, **open-source software** helps make these resources available to everyone. With software such as R and Python, researchers can document their data analysis procedures (in so-called scripts) that others can reuse or adapt.

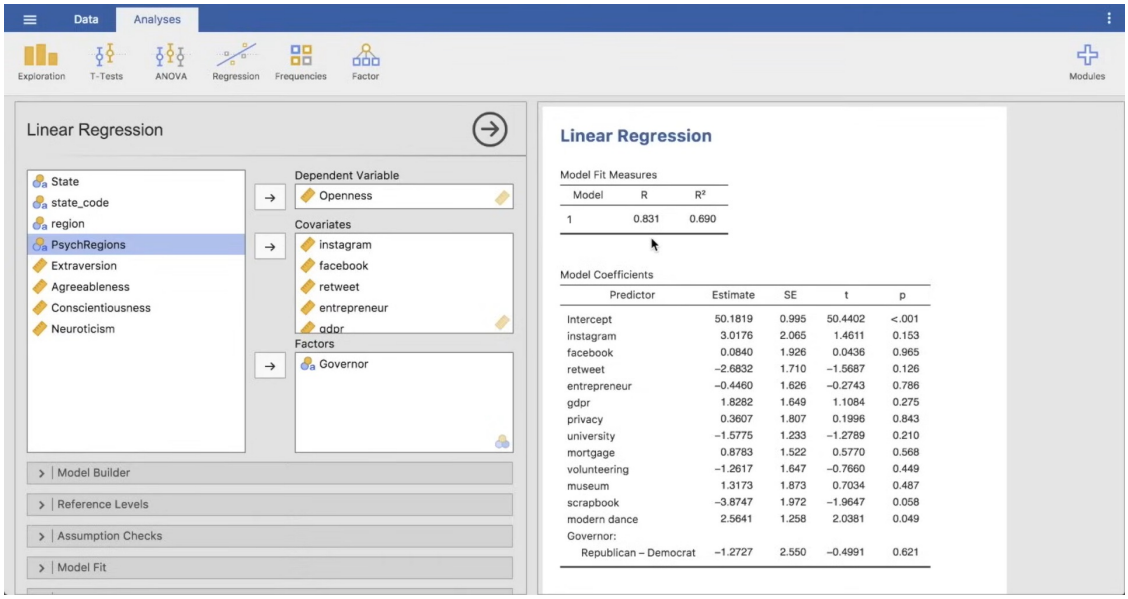
For those less familiar with programming, **point-and-click tools** like JASP, jamovi, and PSPP offer user-friendly alternatives. These free options remove financial barriers and make it easier for others to verify and build on your work. In contrast, proprietary software such as SPSS or MPlus restricts accessibility, limiting reproducibility to users who can afford expensive licenses.

Advantages

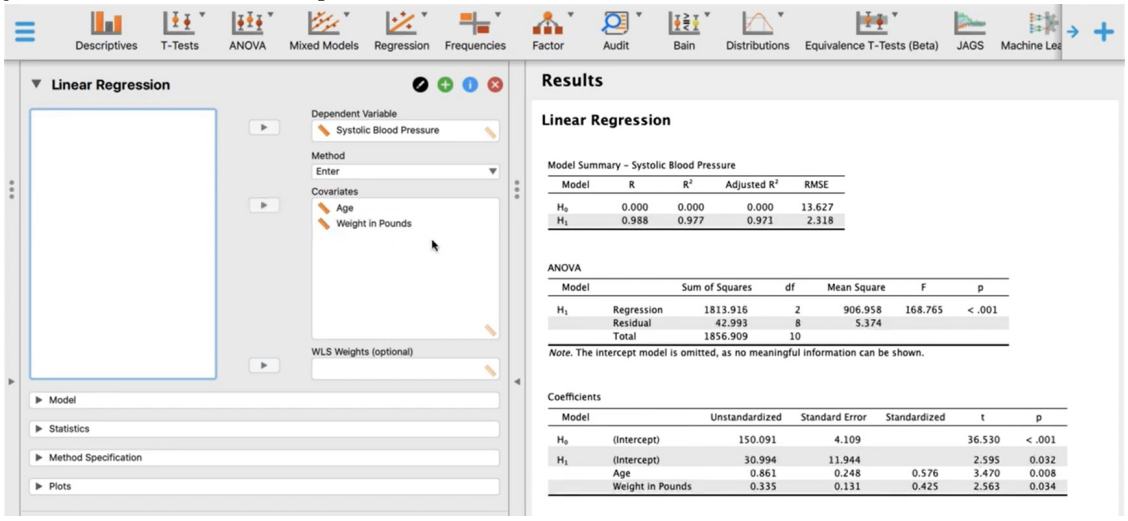
- Fundamentally enables reproducibility, as other researchers can open and run the analyses
- Open-source software lets researchers see how the software processes data when using its functions
- Promotes equity in access to research

First easy steps

Coming from SPSS



jamovi: Mouse over image to zoom



JASP: Mouse over image to zoom

In case your data is already saved and labelled in SPSS: **All** suggested **programs** can **read .sav files** and retain the labelling (such as variable and value labels)!

If you are used to **point-and-click user-interface**: jamovi and JASP are very similar to SPSS in look and feel. This means that there is usually no need for a longer period of

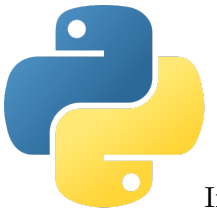
familiarization with the program.
How to share:
jamovi and JASP will automatically create project files that include everything including the data and output of your analyses. Therefore, after loading the data and computing analyses, you can simply share the .jasp (in JASP) or .omv (in jamovi) file with your colleagues or as a supplement to your publication. When others open this file, the data and all the calculated analyses will be available and ready to use in exactly the same way as with you.

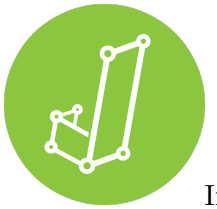
Coming from MPlus, SAS or stata

Learning a new language such as R or python isn't necessarily easily done within a day, it is a **continuous learning process**. The good news is: There are **many free resources** for an introduction to the language (examples see below) and for solving concrete problems you will run into.
Getting help:

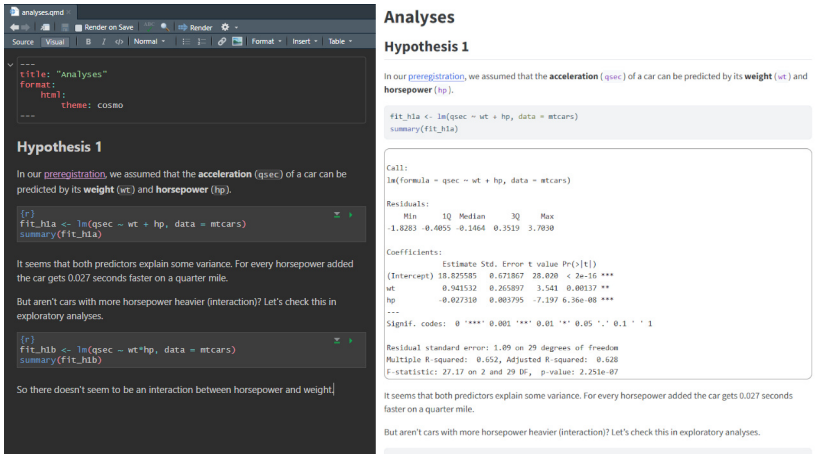
- You will usually find good documentation (or "help") of packages you use or vignettes (examples) of how they can be used. This is an example on how to call and use the R help.
- The Python and R communities are very active online. If you face a problem, chances are someone else has already asked about it. A great place to find solutions is stackoverflow.com.
- Chatbots like ChatGPT are also a fantastic resource for solving problems in your R or Python code. They are trained to provide explanations and suggest fixes to help you tackle your challenges.

Free Resources

 Intro to python(edX online course)  Intro to R(edX online course)


Intro to JASP(youtube: JASP Statistics) 
Intro to jamovi(youtube: freeCodeCamp.org)

2 Input-output-documents ("Notebooks")



Quarto Markdown in R: Mouse over image to zoom

Input-output documents integrate data analysis code (input) with corresponding results (output) in a **unified format**. Tools such as RMarkdown, Quarto Markdown, and Jupyter Notebooks (compatible with R or Python) enable the combination of code, results, and explanatory text—such as interpretations—into a single document exportable as HTML or PDF. Similarly, JASP and jamovi provide built-in functionality to achieve this integration within their platforms.

Advantages

- Provides provenance of results: Directly links transparent inputs to outputs
- Ensures error-free output: HTML/PDF documents are only rendered if all code, from data import to variable manipulation and analysis, runs without errors
- Enhances understandability for others and your future self: Enables detailed explanations of analytical approaches and result interpretations

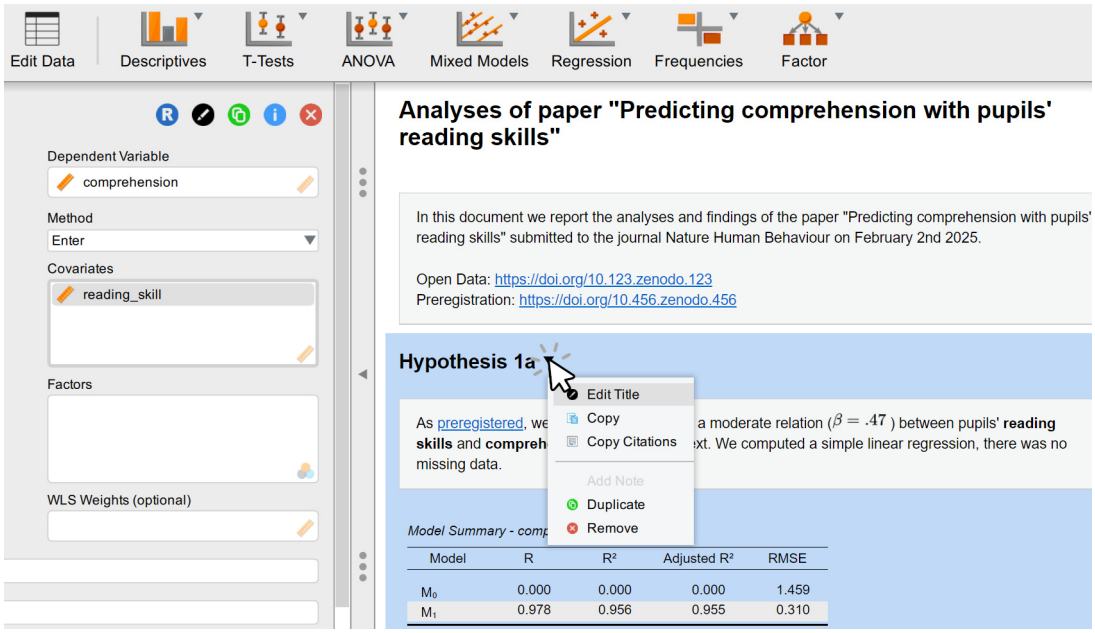
First easy steps

Creating input-output documents requires only **minimal adjustments** to your existing workflow.

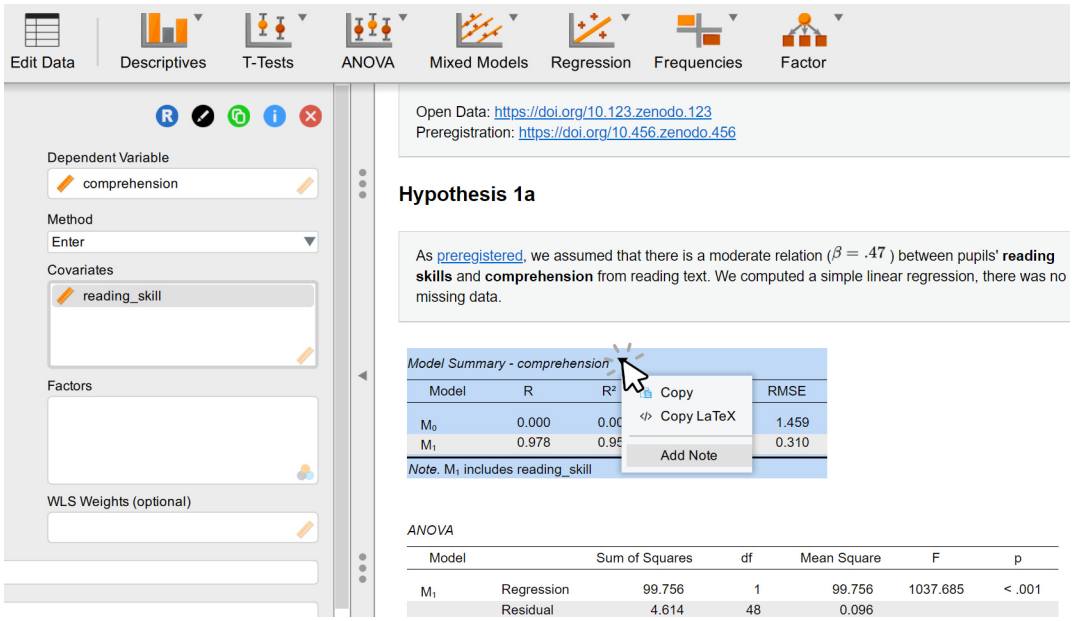
The additional information text you might include to make your data analyses more transparent can vary across projects. However, it can be helpful to communicate the following:

- **Rationale for Decisions**Share the reasoning behind key methodological choices, such as removing outliers, selecting the number of imputed datasets when addressing missing data, or computing specific contrasts. These explanations can make your analysis easier to follow.
- **Connection to Manuscript Sections**Organize your documentation in a way that aligns with the structure of your manuscript. For example, headings like "Treatment Check" or "Hypothesis 1a" can improve readability and help readers navigate the document more efficiently.

With JASP



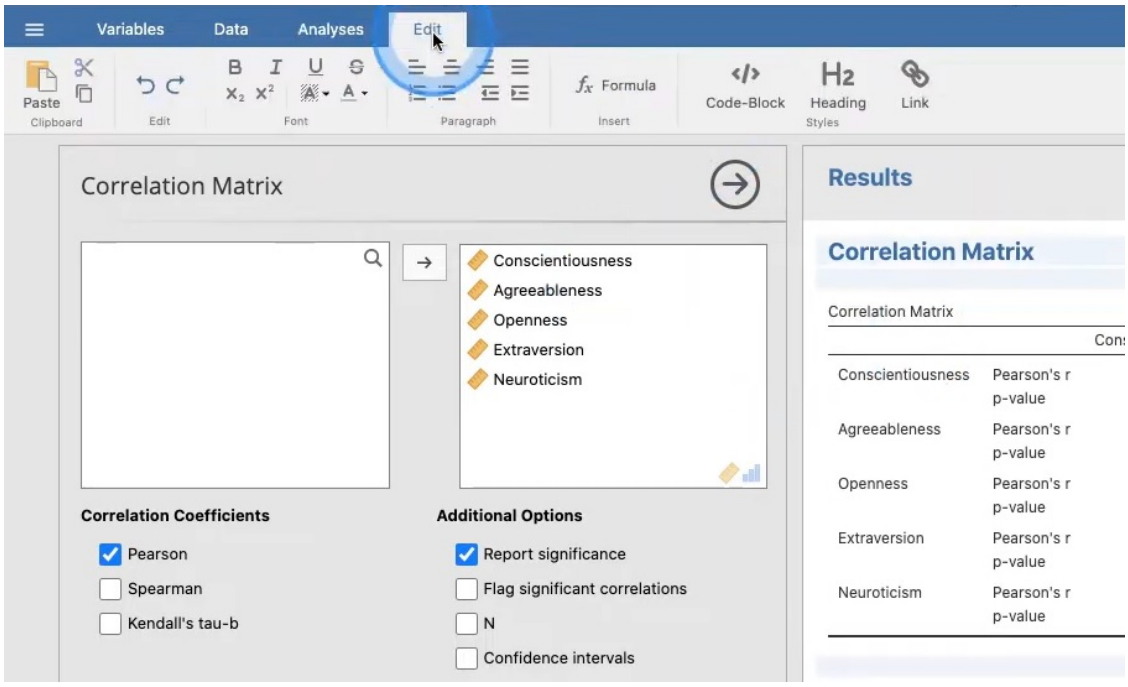
Edit headings and the title by clicking on the down arrow.
Mouse over image to zoom



Add notes to any heading or table by clicking on the down arrow.
Mouse over image to zoom

After you computed you analyses, edited titles and added notes: **Share the .jasp-file** (e.g., on zenodo, as supplement to your manuscript or via email).
This file will **contain everything** from data and analyses to titles and notes in your output!

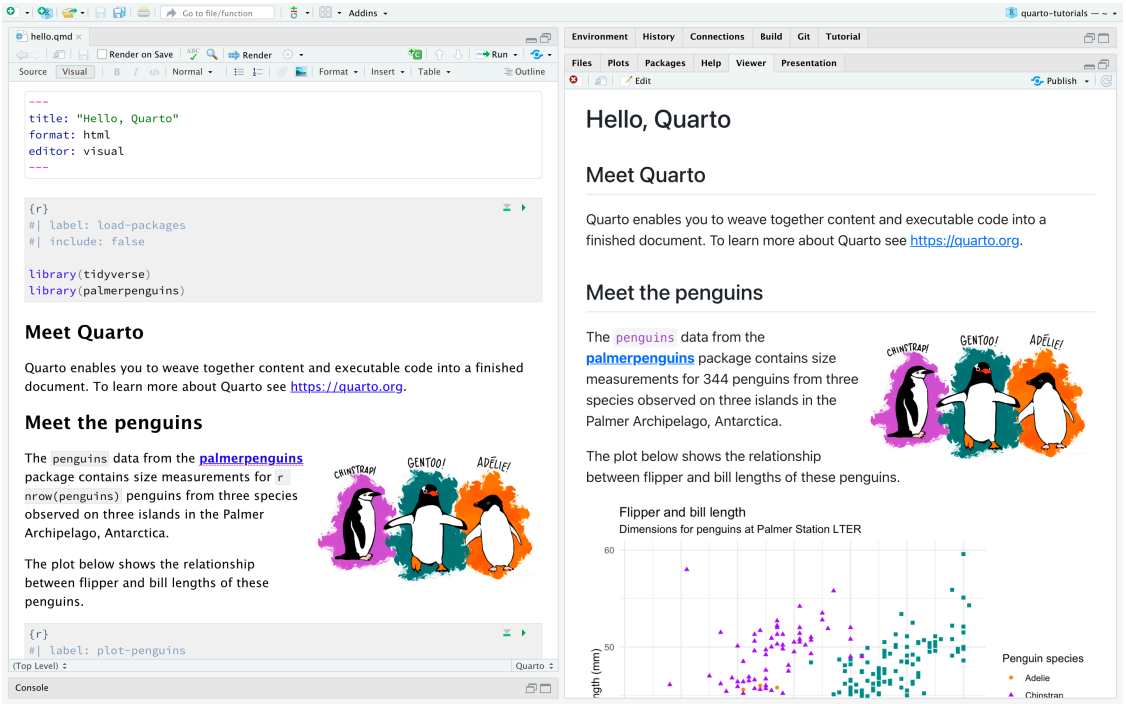
With jamovi



Click on the **edit tab** in the menu on the top, to edit all parts of the output (except results themselves). Change titles to fit the sections of your manuscript and provide further information below titles or around tables and figures.
Mouse over image to zoom

After you computed you analyses, edited titles and added notes: **Share the .omv-file** (e.g., on zenodo, as supplement to your manuscript or via email).
This file will **contain everything** from data and analyses to titles and notes in your output!

With R or python



Mouse over image to zoom: Quarto visual editor (left) and rendered HTML (right)

Quarto is a file format that allows you to provide all your code (from importing to wrangling, transforming and analyzing data) along with (but not limited to) additional text and images.

The advantage of Quarto over RMarkdown is that it works simultaneously with R, Python, Javascript and Julia. It is also **pre-installed in RStudio**: Go to *File* > *New File* > *Quarto Document*.

Quarto has a **visual editor** that works like familiar word processing programs, such as Google Docs or Microsoft Word. In Quarto, you can add special sections called "code chunks" (shown as grey boxes in the picture). These chunks are where you can write or paste R or Python code. If you already have your code in another file, you can just **copy and paste** it into these chunks. You can also add headings and extra text anywhere in the document to explain your code or organize your work.

Finally, **render the entire document** to HTML, PDF or Word by clicking the "render" button.

Free Resources

 "Get started with Quarto"(youtube: Posit PBC)  Share & annotate JASP(youtube: JASP Statistics)

 Share jamovi(youtube: codecamp.org)  Annotate jamovi(youtube: Alexander Swan, Ph.D.)

3 Containerization and version management

R and python use additional extensions ("packages") to expand their basic functions for data analysis. However, differences in the **versions** of these packages can create compatibility issues, even when researchers are using the same version of R of python. Packages like `renv` and `groundhog` for R help to keep these versions consistent.

Beyond package management, maintaining compatibility across **system requirements** — such as operating systems and software versions — is critical for ensuring portability. Containerization tools like Docker and the `holepunch` package for R provide solutions by encapsulating scripts and their dependencies into isolated environments. To what extent JASP and jamovi will be backwards compatible in the future is not entirely clear.

To tackle **both challenges** simultaneously, the `rix` package offers an integrated solution, combining system requirements management with package version control to streamline reproducibility efforts.

Advantages

- Data analysis is sustainable over longer period of time
- Equal system environment between researchers
- Equal package versions between researchers

First easy steps

I recommend **starting with** ensuring the compatibility of **package versions**.

- If you want your workflows and particularly those of others to be altered the least, `groundhog` may be the better option (only minimal alterations with both)
- In case you are using packages installed from github, `renv` may be the even more robust option (both are relatively robust)

Once you are confident in managing packages, you can proceed to address system requirements, such as standardizing software versions. See "Free resources" for `holepunch` and `rix` below.

Package management with renv (for R)

You

1. **Only once:**
   Install `renv` by running

```
install.packages("renv")
```

2. **Once per project:**
   Make sure you work within an R-project (and not with single files)!
   Initiate `renv` by running

```
renv::init()
```

3. **Only in case you installed new packages:**
   Document these new packages by running

```
renv::snapshot()
```

4. **Now share your project with others** (e.g. via Zenodo, GitHub, email)

Others

1. Install `renv` by running

```
install.packages("renv")
```

2. Open the project they received from you
3. To have the same package versions available, others need to run

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

Package management with groundhog (for R)

1. **Only once:**
   Install `groundhog` by running

```
install.packages("groundhog")
```

2. **When starting RStudio:**
   Load `groundhog` package by running

```
library(groundhog)
```

3. **When writing a script:**

You can replace this

```
library(rio)
library(dplyr)
```

with this using e.g. today's date

```
groundhog.library("
  library(rio)
  library(dplyr)   ", "2025-01-28")
```

In contrast to `renv`, **others do not need to know anything** about `groundhog`. They can run your script, that will automatically install and load the specific packages.

Free Resources

Tutorial: renv

Tutorial: groundhog

Tutorial: Holepunch

Tutorial: rix

Foto by Marten Bjork on Unsplash