

Computational Reproducibility

Jürgen Schneider

2025-01-22

Table of contents

General	3
Contents	3
1 Open file formats and software	4
Advantages	4
First easy steps	5
Free Resources	6
2 Input-output-documents (“Notebooks”)	8
Advantages	8
First easy steps	9
Free Resources	13
3 Containerization and version management	14
Advantages	15
First easy steps	15
Free Resources	17

General

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

- 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

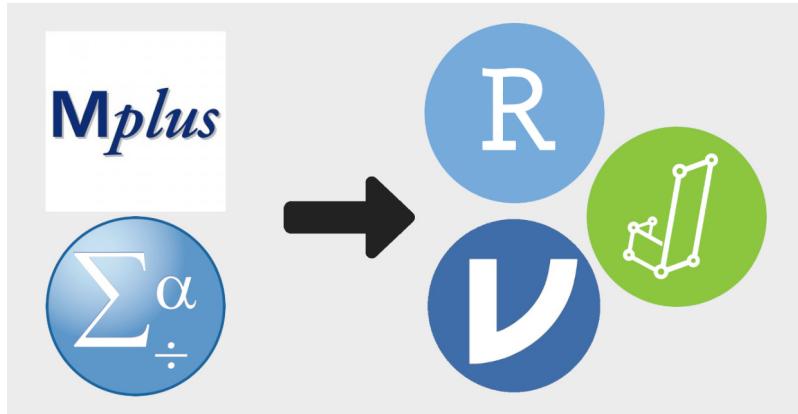
The [Workflow for Open Reproducible Code in Science \(WORCS\)](#) is an excellent framework that also integrates recommendations we give here.

Contents

See also navigation menu on the left.

Open file formats and software
Ensuring cost-free access to your analyses
Input-output-documents
Making analyses comprehensible for yourself and others
Containerization and version management
Making 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

Model	R	R ²
1	0.831	0.690

Predictor	Estimate	SE	t	p
Intercept	50.1819	0.995	50.4402	<.001
instagram	3.0176	2.065	1.4611	0.153
facebook	0.0840	1.926	0.0436	0.965
retweet	-2.6832	1.710	-1.5687	0.126
entrepreneur	-0.4460	1.626	-0.2743	0.786
gdpr	1.8282	1.649	1.1084	0.275
privacy	0.3607	1.807	0.1996	0.843
university	-1.5775	1.233	-1.2789	0.210
mortgage	0.8783	1.522	0.5770	0.568
volunteering	-1.2617	1.647	-0.7660	0.449
museum	3.1373	1.873	0.7034	0.487
scrapbook	-3.8747	1.972	-1.9647	0.058
modern dance	2.5641	1.258	2.0381	0.049
Governor:				
Republican – Democrat	-1.2727	2.550	-0.4991	0.621

jamovi: Mouse over image to zoom

Model	R	R ²	Adjusted R ²	RMSE
H ₀	0.000	0.000	0.000	13.627
H ₁	0.988	0.977	0.971	2.318

Model	Sum of Squares	df	Mean Square	F	p
H ₁ Regression	1813.916	2	906.958	168.765	<.001
Residual	42.993	8	5.374		
Total	1856.909	10			

Model	Unstandardized	Standard Error	Standardized	t	p
H ₀ (Intercept)	150.091	4.109		36.530	<.001
H ₁ (Intercept)	30.994	11.944		2.595	0.032
Age	0.861	0.248	0.576	3.470	0.008
Weight in Pounds	0.335	0.131	0.425	2.563	0.034

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.

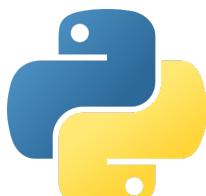
Coming from MPlus, SAS or stata

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)
course)



Intro to R(edX online



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



Intro to

2 Input-output-documents (“Notebooks”)

The screenshot shows a Quarto Markdown in R notebook titled "Analyses". It contains two sections: "Hypothesis 1" and "Analyses".

Hypothesis 1:

```
[r] fit_hia <- lm(qsec ~ wt + hp, data = mtcars)
summary(fit_hia)
```

It seems that both predictors explain some variance. For every horsepower added the car gets 0.027 seconds faster on a quarter mile.

But aren't cars with more horsepower heavier (interaction)? Let's check this in exploratory analyses.

```
[r] fit_hib <- lm(qsec ~ wt+hp, data = mtcars)
summary(fit_hib)
```

So there doesn't seem to be an interaction between horsepower and weight.

Analyses:

In our pre-registration, we assumed that the acceleration (qsec) of a car can be predicted by its weight (wt) and horsepower (hp).

```
fit_hia <- lm(qsec ~ wt + hp, data = mtcars)
summary(fit_hia)
```

Call:

```
lm(formula = qsec ~ wt + hp, data = mtcars)
```

Residuals:

Min	IQ	Median	3Q	Max
-1.4283	-0.4055	0.1464	0.3519	3.7838

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	18.835585	0.671967	28.098	<2e-16 ***
wt	0.941532	0.265997	3.541	0.00137 **
hp	-0.027318	0.003955	-7.197	6.36e-08 ***

Signif. codes:	0 ****	0.001 ***	0.01 **	0.05 *

Residual standard error: 1.89 on 29 degrees of freedom
Multiple R-squared: 0.652, Adjusted R-squared: 0.628
F-statistic: 27.17 on 2 and 29 DF, p-value: 2.251e-07

It seems that both predictors explain some variance. For every horsepower added the car gets 0.027 seconds faster on a quarter mile.

But aren't cars with more horsepower heavier (interaction)? Let's check this in exploratory analyses.

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

Analyses of paper "Predicting comprehension with pupils' reading skills"

In this document we report the analyses and findings of the paper "Predicting comprehension with pupils' reading skills" submitted to the journal Nature Human Behaviour on February 2nd 2025.

Open Data: <https://doi.org/10.123.zenodo.123>
Preregistration: <https://doi.org/10.456.zenodo.456>

Hypothesis 1a

As [preregistered](#), we found a moderate relation ($\beta = .47$) between pupils' reading skills and comprehension. We computed a simple linear regression, there was no missing data.

[Edit Title](#)
[Copy](#) [Copy Citations](#)
[Add Note](#)
[Duplicate](#) [Remove](#)

Model	R	R ²	Adjusted R ²	RMSE
M ₀	0.000	0.000	0.000	1.459
M ₁	0.978	0.956	0.955	0.310

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

Open Data: <https://doi.org/10.123.zenodo.123>
Preregistration: <https://doi.org/10.456.zenodo.456>

Hypothesis 1a

As [preregistered](#), we assumed that there is a moderate relation ($\beta = .47$) between pupils' **reading skills** and **comprehension** from reading text. We computed a simple linear regression, there was no missing data.

Model	R	R ²	RMSE
M ₀	0.000	0.00	1.459
M ₁	0.978	0.95	0.310

Note. M₁ includes reading_skill

Model	Sum of Squares	df	Mean Square	F	p
M ₁	Regression	99.756	1	99.756	1037.685 < .001
	Residual	4.614	48	0.096	

Add notes to any heading or table by clicking on the down arrow.

Mouse over image to zoom

After you computed your 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

The screenshot shows the jamovi software interface. The top menu bar includes 'Variables', 'Data', 'Analyses', 'Edit' (which is highlighted), 'Clipboard', 'Edit', 'Font', 'Paragraph', 'Insert', 'Code-Block', 'Heading Styles', and 'Link'. The main window displays a 'Correlation Matrix' dialog box. On the left, there's a search icon and a list of variables: Conscientiousness, Agreeableness, Openness, Extraversion, and Neuroticism. Below this is a section titled 'Correlation Coefficients' with checkboxes for Pearson (checked), Spearman, and Kendall's tau-b. To the right is an 'Additional Options' section with checkboxes for Report significance (checked), Flag significant correlations, N, and Confidence intervals. On the far right, the 'Results' panel shows a 'Correlation Matrix' table with Pearson's r and p-value for each pair of variables.

	Conscientiousness	Agreeableness	Openness	Extraversion	Neuroticism
Conscientiousness	Pearson's r p-value				
Agreeableness		Pearson's r p-value			
Openness			Pearson's r p-value		
Extraversion				Pearson's r p-value	
Neuroticism					Pearson's r p-value

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 your 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

The screenshot shows the Quarto visual editor interface on the left and the resulting rendered HTML document on the right.

Quarto Visual Editor (Left):

- The file is named "hello.qmd".
- The "Source" tab is selected, showing the Quarto code:

```
---
```

```
title: "Hello, Quarto"
format: html
editor: visual
---
```

```
{r}
#| label: load-packages
#| include: false

library(tidyverse)
library(palmerpenguins)
```

- A "Meet Quarto" section contains text and a penguin illustration labeled "CHINSTRAP", "GENTOO!", and "ADELIE!".
- A "Meet the penguins" section contains text and a penguin illustration.
- A "Plot-penguins" code chunk is shown in the console.

Rendered HTML (Right):

- The title is "Hello, Quarto".
- A "Meet Quarto" section with explanatory text and a link to <https://quarto.org>.
- A "Meet the penguins" section with explanatory text and a penguin illustration.
- A scatter plot titled "Flipper and bill length Dimensions for penguins at Palmer Station LTER". The y-axis is "Bill length (mm)" ranging from 50 to 60. The x-axis is "Flipper length (mm)" ranging from 50 to 60. Data points are colored by species: Adelie (orange), Gentoo (teal), and Chinstrap (purple).

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)
annotate JASP(youtube: JASP Statistics)



Share & an-



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



Annotate jamovi(youtube:

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 or 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](#) [groundhog](#) Tutorial: [groundhog](#) Tutorial:

R <- R



Holepunch Tutorial: [rix](#)

Foto by Marten Bjork on Unsplash