Reproducible Documents

Why do we need reproducible documents?

Traditional workflows within science use many different kinds of tools. This requires careful recording of what happens at each step of the scientific process. Good laboratory notebook practices are a feature of many sciences but are not typically taught as part of a psychology curriculum. This combination within psychology of a lack of clear recording of data collection and analysis practices can lead to difficulties in retracing the steps that led to any given outcome in science. A traditional workflow in psychology may involve the design of research and preparation for ethical review using a word processor and perhaps some diagramming tools. This is followed by data collection on a different platform perhaps using pen and paper in a laboratory, but increasingly using automated platforms such as Qualtrics, PsychoPy, or jsPsych. Following this data is manipulated from the shape it comes from the platform and is wrangled into the right shape for analysis sometimes in a statistical program, but often using cut and paste techniques in Microsoft Excel or a similar spreadsheet program. The data is then analysed using some statistical software platform such SPSS, SAS, JASP, Jamovi or R. When the results are found these are extracted and the whole sequence is written up using some combination of a word processor—typically Microsoft Word—and perhaps some citation software such as Endnote or RefWorks.

The problem with this approach is that it is not very reproducible, the steps are difficult to trace unless the researcher is inordinately meticulous with their recording and it is prone to untraceable errors.

The solution is to create a reproducible document.

What are reproducible documents?

Reproducible documents are a way of eliminating as many of these disparate steps as possible and maximising the recording of the steps that are taken along the research process. The goal is to streamline the research process while making it more transparent and reproducible. Ideally these documents will be made available for the scrutiny of other scientific researchers and the public, but there are many advantages to using reproducible documents as they make the process of conducting science more efficient and they make research much easier to pick up after the inevitable six month to year hiatuses that occur throughout the scientific process as

projects go through their different stages and the myriad competing attention drawing factors of a professional researchers life fragment our time.

We are not yet at a stage where all of the various platforms and processes can be accumulated into one work flow but we can do much better than the traditional approach.

Typically a reproducible document is a combination of some word processing tools to create the document with mechanisms to insert computer code chunks that allow us to load, store, and manipulate data; conduct statistical and mathematical analysis; visualise the analysis producing tables and plots as part of the document output; and integrating with citation software to allow the easy creation of references.

The tools of reproducible documents

Document preparation tools

There are a range of preexisting computational technologies that can be brought together to enable these documents. The TeX and LaTeX systems are well established document preparation and typesetting systems (latex-project), these are aided by the use of Markdown (markdown-guide.org) which provides a very simple document markup language which the researcher uses to create the documents and pandoc (pandoc.org) which allows the same markdown document to be output to many different formats. One markdown document can be used to create websites, PDFs, Word documents, ePub books and many more. These represent the word processing part.

Code chunks

The computer chunks are comprised of various different languages that can be mixed and matched, the most commonly used languages are R and Python, but languages like Julia, Stan, Javascript, SQL, C, and Fortran.

The current document is being written as a reproducible document using the RStudio software and I can include a code chunk in R

```
data <- c(1,2,3,4,5,6,7,8,9,10)
x <- mean(data)
print(x)
```

```
[1] 5.5
```

or in python

```
import statistics
data = [1,2,3,4,5,6,7,8,9,10]
x = statistics.mean(data)
print(x)
```

```
5.5
```

Code chunks can be hidden and only the output produced which is useful when including any data wrangling code that you do not want to be seen in a finished output, of for creating tables and plots. The important thing is that the code remains visible in the source form of the document so it can be inspected if necessary and it stays in the one place with the end product document. You can see this process at work in Figure 1, which contains code included in the source document but hidden in the final document.

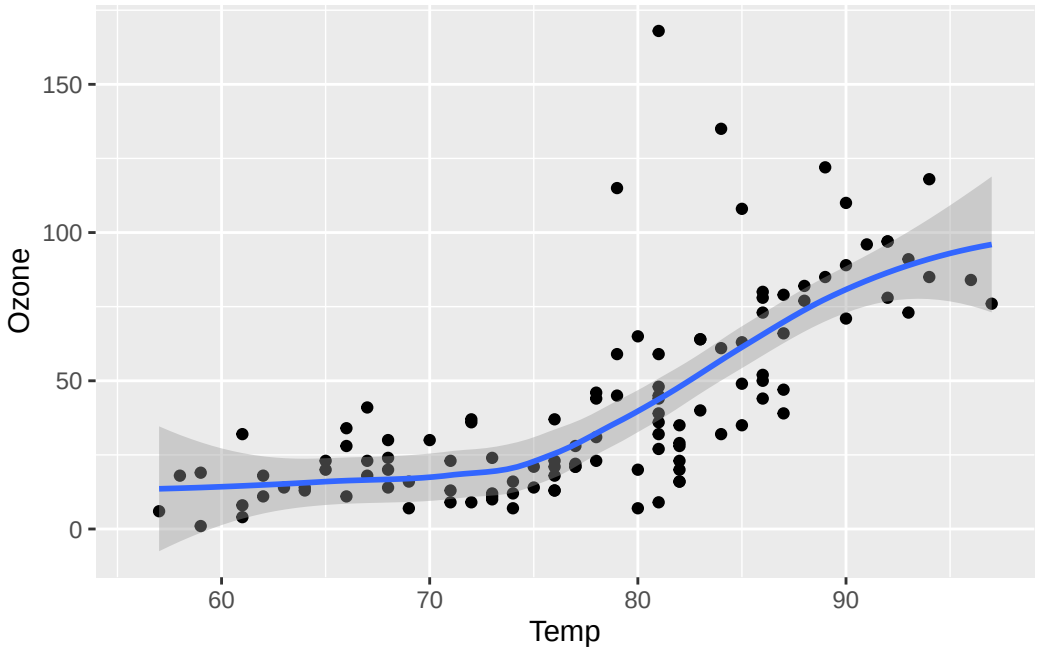

Figure 1: Temperature and ozone level.

Citation software

Citation management and the automatic creation of a references section are important components of reproducible document systems. Most systems interact with pandoc, BibTeX, Zotero, Crossref, PubMed, and the DOI system to allow in text citations with automated References sections (e.g. McKeown, 2013). This reference will appear in the References section automatically without any more work.

Version control

An important but optional companion technology to the creation of reproducible documents is the variety of version control software such as git or svn. Git is the currently most popular type of version control. Version control systems were developed to keep track of the various changes that were made to computer code and to allow variations of code to be made and merged into a single document and to allow collaboration by multiple coders without the danger of overwriting each other. This turns out to be handy when it comes to writing documents too, especially documents that contain code. Think of version control as a high powered and rigorous combination of "track changes" and backing up files. It is a solution to folders full of final final final versions of the same document. Git is a piece of software that sits on your local computer and controls the versions of your documents, but it has a companion in github that offers remote hosting of documents and enables collaboration between multiple contributors. You can see the code for this document at (github.com/QUBPsy/QUBPsyOpenScience).

Reproducible Document Platforms

There are a variety of ways of creating reproducible documents. Here we will look at some of the options.

RStudio and R Markdown

One of the two major pioneers of mechanisms to produce reproducible documents came from the combination of the R focused Integrated Development Environment (IDE) RStudio. RStudio started in 2011 to create an easy environment in which to write R code and install and manage R packages. In 2012 Yihui Xie introduced the knitr package to enable the creation of documents with code chunks embedded in them. Markdown become the most popular language to develop these documents and the rmarkdown package was introduced in 2014, and it became well embedded within the RSTudio IDE where a lot of effort was made to make the production of RMarkdown documents simple and easy. The kinds of documents that could be produced in RMarkdown, webpages and websites, pdf documents, Shiny web apps, journal articles, slides and presentations, and books.

The R Markdown ecosystem is now quite mature and there are many packages that allow people to get up and running with many different aspects such as: the creation of quite sophisticated websites using RStudio and Github pages (with many other options); the creation of APA style manuscripts and journal articles using papaja (although this remains somewhat buggy); templates to aid in the creation of many different kinds of journal articles through rticles

such as Elsevier, IEEE, Royal Society, Springer, PNAS journal templates (Table 1 provides a full list of journal abbreviations); full book creation in online, pdf and ePub formats using bookdown (see an example here); various kinds of presentations (Powerpoint, Beamer, Slidy); dashboards using the flexdashboard package; and interactive documents incorporating Shiny web apps and html widgets.

Table 1: The abbreviations for the R Markdown journal article templates avaialble in the rticles R package.

acm	frontiers	oup_v0
acs	glossa	oup_v1
aea	ieee	peerj
agu	ims	pihph
ajs	informs	plos
amq	iop	pnas
ams	isba	rjournal
arxiv	jasa	rsos
asa	jedm	rss
bioinformatics	joss	sage
biometrics	jss	sim
copernicus	lipics	springer
ctex	lncs	tf
elsevier	mdpi	trb

One issue that arises with reproducible documents arises due to their instrinsic nature; they contain code. Code depends on other code and code changes with time. There are constant updates to packages and software and often these updates will break older code as functions become deprecated or change in nature. The solution to this problem is to create environments in which to encapsulate code and its dependencies. In R the Packrat and newer renv packages are designed to provide this dependency management within R. These packages will create a local environment snapshot of all the necessary code and allow them to reloaded whent he time comes to work with the files. An example of a reproducible document that uses renv to do this can be found at this github repository which provided the code to produce and run all the experiments need to produce this paper (Dupré et al., 2020).

JupyterLab, Jupyter Notebooks and Python

Another option for creating reproducible documents that has its origins in the Python computer language ecosystem is Jupyter Notebooks. Typically these require knowledge of the

Python programming language and comfort working in a command line environment such as terminals in Unix environments and Windows Terminal in Windows 10 and above (previously the Windows Console).

A Jupyter Notebook is a reproducible document that normally sits in a folder on your computer and you interact with using an interface in your browser. You navigate to the appropriate folder in your terminal and type "jupyter notebook" and this opens your browser and browser window that allows you to interact with the document. Documents contain a mix of text and code. The is a classic Jupyter Notebook interface and a more advanced IDE style interface known as JupyterLab. There are online servers that will allow you to interact with Jupyter Notebooks without the need to install software and have a dedicated machine, some of these can be seen at the Jupyter project: classic interface, JupyterLab interface. Similarly to RStudio even though they started firmly focused on one language—Python—but they also realised the value in supporting multiple languages and allowing them to be used interchangeably, they now support about 40 languages including R (R in a Jupyter Notebook).

Dependency issues in Python can be dealt with in a similar way to R by creating environments in which the code and packages can be isolated from other environments. The use of virtual environments in Python is more well established than in R and there are a variety of ways to create them. For example, the venv module, the virtualenv package, or within the Anaconda Python distribution platform. Virtual environments are often advised as best practice within general Python usage.

Quarto

The two previous reproducible document approaches showed a trend from focusing on a single language towards reproducible documents that are more flexible. Recently the people at RStudio have announced that they have taken this a step further with the development of Quarto, Quarto is a reproducible document system again built on Pandoc allowing documents to be created using Markdown or Jupyter Notebooks. Quarto is installed independently of the other systems as separate software on the computer and then each of the systems interface with it. Reproducible documents can then be created in a text editor, RStudio, Jupyter Notebooks or VS Code (Microsoft's Visual Studio Code IDE). The goal is to make the document production part of the system agnostic to the way in which people prefer to work. As it is being spearheaded by RStudio it is likely to supersede the current R Markdown ecosystem (although R Markdown will be around for a long time). In producing Quarto the Rstudio team have taken the opportunity to make some of the issues that existed with R Markdown better and they seek to make a more unified system for developing each of the different styles of output. It is still in its infancy but I expect that it will quickly become the standard within the R ecosystem

at least. This current book/website hybrid has been created using Quarto, although it's very first incarnation was as a R Markdown Bookdown project.

VS Code

Microsoft's Visual Studio Code is an IDE that is used by many people who do development within the Windows ecosystem. With the arrival of Quarto it can now be used as an environment for developing reproducible documents. It requires the addition of a Code Extension. This may be useful if you are already a user of VS Code otherwise the Rstudio IDE is probably a better place to start.

Dupré, D., Krumhuber, E. G., Küster, D., & McKeown, G. J. (2020). A performance comparison of eight commercially available automatic classifiers for facial affect recognition. *PLOS ONE*, *15*(4), e0231968. https://doi.org/10.1371/journal.pone.0231968

McKeown, G. J. (2013). The Analogical Peacock Hypothesis: The Sexual Selection of Mind-Reading and Relational Cognition in Human Communication. *Review of General Psychology*, *17*(3), 267–287. https://doi.org/10.1037/a0032631