# Reproducible Analytical Pipelines

<table>
<thead>
<tr>
<th>Document Control</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
</tr>
<tr>
<td><strong>Date Issued</strong></td>
</tr>
<tr>
<td><strong>Author</strong></td>
</tr>
<tr>
<td><strong>Comments to</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>December 2018</td>
<td>First draft</td>
<td>Anna Price</td>
</tr>
<tr>
<td>Version 1.1</td>
<td>December 2018</td>
<td>Edited following comments on first draft from David Caldwell, Esther Morris and Jack Hannah</td>
<td>Anna Price</td>
</tr>
<tr>
<td>Version 1.2</td>
<td>January 2019</td>
<td>Edited following comments from SAG members</td>
<td>Anna Price</td>
</tr>
<tr>
<td>Version 1.3</td>
<td>February 2019</td>
<td>Final draft following minor comments from SAG members</td>
<td>Anna Price</td>
</tr>
<tr>
<td>Version 1.4</td>
<td>February 2019</td>
<td>Addition of new flow charts created by graphics team and updated table</td>
<td>Anna Price</td>
</tr>
</tbody>
</table>
Introduction

Reproducible research is the idea that academic studies and experiments should be presented in such a way that others can verify results and expand on them. The concept has been well-recognised and encouraged within academia over the last ten to fifteen years, and is now relevant, if not integral, to data science. Although focused in the academic community and across industry, more recently the concept of reproducible analysis has been combined with the desire to automate statistical publication production in the public sector. This resulted in the Reproducible Analytical Pipeline which carries out all the steps of the publication process in one open source software program, from data extraction to report production. It aims to improve the quality, auditability and speed of publication production, as well as ensure knowledge transfer in organisations with high turnover in staff.

It is important to recognise that Reproducible Analytical Pipelines will not be suitable for every publication. However implementing just one or two techniques such as version control or peer review has the potential to improve quality and make efficiency savings (the incremental levels of Reproducible Analytical Pipelines are outlined later in this paper in Table 1 and discussed in Recommendations and limitations). Furthermore, many of these concepts are applicable to any piece of analytical work, not just statistical publications or regularly produced reports.

Reproducible analysis

Reproducible analysis provides evidence of the correctness of results, enables others to make use of methods and results, and exposes workflow to others. All of these benefits apply within and across teams at ISD. They are also relevant to the Trustworthiness pillar in the UK Statistics Authority Code of Practice because the transparency required for work to be reproducible creates trust among colleagues and peers, with other producers of statistics, and with the wider general public.

In order to determine whether work is reproducible, some key questions\(^1\) to ask are:

- Is it clear where to begin?
- Can you determine which inputs produced which outputs?
- Which is the most recent copy of the code?
- What files can I safely delete, and should I delete files at all?
- Are scripts and reports version controlled?
- Are there lots of manual steps in the process?
- Are proprietary software used in the process?

For example, every time a ‘point and click’ or ‘copy and paste’ step is required, there is a lack of reproducibility as this process is not documented in code. Use of proprietary software also reduces reproducibility. Externally, users cannot reproduce work that was carried out using proprietary software if they do not have the required license. Internally we also risk not being able to reproduce previous work if we stop using certain proprietary software in future.

In order to help with reproducibility, modern computing methods and software can be utilised. To write and publish self-contained documents that include both narrative and code, a feature of R called R Markdown can be used. Version control software such as git and cloud-based code repositories such as GitHub allow changes to code to be recorded and previous versions of files to be recalled later. This automatically builds documentation into the publication production process and creates a record of what has been done and why which can be audited in future. Finally, software such as Docker allows entire computing environments to be shared among colleagues with dependencies kept intact. This means that you can save the computing environment as it was

---

\(^1\) A checklist of further questions regarding reproducibility
when a publication was run, and reproduce the publication with exactly the same software and versions of software in the future.

### The traditional publication process

Figure 1 shows an example of the traditional publication process, not just within PHI but across the public sector. Data are extracted from a database or data file and analysis is carried out in proprietary software. The results of this analysis are moved to spreadsheets and checked using Excel. The final results are then copied and pasted from Excel into a Word document containing the previous publication text and figures. Other content within the Word document, such as dates, are manually updated. Finally, this Word document is transformed into a PDF for release.

![Example of the traditional publication process within PHI](image)

The process may be even more complex than the one presented above if more than one person is involved in each step or if the quality assurance loop needs to be carried out multiple times when errors in the original data are found and need to be fixed.

There are a number of problems with this traditional method of producing publications. The complex process involves moving data from one proprietary software to another which, as discussed above, reduces reproducibility. It also increases the risk of manual error, as does the requirement for a number of manual 'copy and paste' steps. The reliance on spreadsheets introduces further risk of errors, since spreadsheets are notorious for containing mistakes (Panko, 2016). There is an issue of knowledge legacy in some teams, where the knowledge of how to produce a publication sits with one or two key team members. Finally, the process requires manual, menial tasks to be carried out by highly skilled analysts.

### Reproducible Analytical Pipelines

In order to address the problems above, the concept of a Reproducible Analytical Pipeline was created by Matthew Upson and Matthew Gregory at the Government Digital Service. They combined the idea of reproducible research with best practice and tools from data science. Fundamentally, a Reproducible Analytical Pipeline (RAP) carries out all the steps of the analytical process in one open source software program, from the extraction of data (preferably from a database), to the data preparation and quality assurance (QA), to the final report (see Figure 2).

Although RAP is language agnostic, the Transforming Publishing team have chosen to use R due to the features of R which lend themselves to reproducible analysis and the knowledge and skills within the team and more widely across PHI.
Figure 2: Desired state of publication process using RAP

A Reproducible Analytical Pipeline should be:

1. **Reproducible**
   This is achieved through the use of open source software, R Projects, package management and ideally computing environment management.

2. **High quality**
   This is achieved by removing manual steps in the production process which increase the risk of errors and incorporating data QA and unit testing of R functions into the pipeline.

3. **Auditable**
   This is achieved through the use of version control and documentation of R functions and code.

4. **Sustainable**
   This is achieved through training and up skilling of staff – “sustainability is social, not technical” (Matthew Upson, 2018) and requires everyone in a team to know how to use and update the pipeline.

Figure 3 shows an example of the full RAP process using R as the primary software:

Figure 3: Detailed example of a Reproducible Analytical Pipeline

Reproducibility and quality of pipeline ensured via: RStudio project; git for version control; GitHub for code repository and peer review; unit testing of functions: R package management; and computing environment control.
Functions are written in order to extract data from a data store and carry out the cleaning, manipulation and analysis required to create the minimal tidy data set (Wickham, 2014). A minimal tidy dataset is the minimum amount of data required to complete the publication. Further R functions are then written to extract elements of the minimal tidy dataset, such as dates, hospital names or numerical values, and automatically populate the R Markdown report. Note that all these steps are contained within one R project or package, so are not the same as the transitions between software in Figure 1. Reproducibility is built into the RAP process by removing manual point-click steps and making use of git and GitHub to create an audit trail and share files. The pipeline is quality assured by writing unit tests (represented in Figure 3 by Travis) which validate that the output of an R function is as expected.

Although the final publication report (and any additional output such as tables or dashboards) will be created automatically via the pipeline, some level of human input will always be required during the production process – for example, final proof reading of the report and checking that any automated text still matches the figures. Changes to the text within a report can be made easily by editing the R Markdown code as and when required.

**Should your publication be automated?**

Deciding whether to automate a publication requires a balance between ease of maintenance, development time required and level of automation. Some questions to ask before embarking on a RAP project:

- How does the department and public benefit?
- How many reports do your team produce?
- What proportion of time is spent on producing reports?
- What would your team do if their time was freed up from menial, manual tasks?
- How much copying and pasting or data movement between software is involved?
- What would the impact of mistakes in production be, e.g. on public policy or funding allocation?
- Could your team create the report if certain team members suddenly left?
- Could you reproduce your publication statistics from 5 years ago?
- What proportion of your spreadsheets or reports have been found to contain errors?

If you feel that your publication would benefit from RAP after discussing the questions above and reviewing Table 1 in your team then please contact the Transforming Publishing team. You should also review the Checklist for TPP and RAP processes, which details how the Transforming Publishing team work and explains what to expect when carrying out a RAP project.

---

2 Questions based on the RAP companion
## Levels of code maturity and automation

### Table 1: Checklist for levels of code maturity and automation

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Data files produced by code (e.g. using R)</th>
<th>Outputs produced by code (e.g. reports and tables made in R)</th>
<th>Minimal tidy data saved with code (e.g. using an R project)</th>
<th>Version control of code</th>
<th>Peer review of code</th>
<th>Automated quality assurance of data</th>
<th>R packages managed</th>
<th>Unit testing of functions</th>
<th>Unit testing guaranteed (e.g. by Travis)</th>
<th>Documentation of functions</th>
<th>Documentation guaranteed</th>
<th>Reproducible computing environment (e.g. using Docker)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ad hoc R code</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>2</td>
<td>R project</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>3</td>
<td>R project under version control (VC)</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>4a</td>
<td>R project under VC and peer reviewed (wrangling)</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>4b</td>
<td>Replicable report in R Markdown (output)</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>5</td>
<td>Near RAP</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>6</td>
<td>Full RAP</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>7</td>
<td>R package</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
**Recommendations and limitations**

Based on the information above, there are a number of recommendations which are made by the Transforming Publishing team:

1. Although Table 1 refers specifically to publications, the same principles can be applied to any piece of analytical work which is produced regularly.

2. Regardless of whether it is a RAP project, we recommend that the minimum standard for all R code in PHI should be level 2 (code and data saved in an R project). If an internal code repository is in place and version control training is widely available to analysts then we recommend the minimum standard should be level 3 (R project under version control).

3. To consider a pipeline reproducible, we recommend that the minimum standard for a RAP project should be level 4 (a or b, or both).

4. Not every RAP project needs to have an R package (level 7) as the end goal. R packages are the “fundamental unit of shareable code” (Hadley Wickham, 2015) and provide conventions which require that code, data, documentation and tests are organised together in a systematic way. However, the desire for an R package will be influenced by the R skills within the team building it. The most important features of reproducible analyses are unit testing, documentation and package and environment management, which do not require the end product to be an R package. We recommend that as a first step a RAP is created as an R project (levels 4-6), and subsequently may be bundled into a package.

It should be noted that the development of RAPs, and the level of automation chosen by a team developing a RAP, is directly affected by the available IT infrastructure and analytical tools strategy within PHI. For example, there is no internally hosted code repository which can be accessed by all analysts (e.g. Gitea, Gogs or GitLab). Although GitHub.com can be used as a code store, private repositories are not free for more than three collaborators. More importantly, data cannot be stored on GitHub.com, even on private repositories because the content is hosted on an Amazon server. This means that data and code cannot be saved together, an important feature of RAP. To ensure fully reproducible results, levels 6 and 7 in Table 1 should include computer environment management (e.g. using Docker). However, such a system is not yet available within PHI.

Furthermore, there is no organisational-wide strategy for managing R packages. The Transforming Publishing team have tested the use of packrat for package management, but found it did not work adequately and caused problems when sharing work among colleagues. Therefore, we do not recommend using packrat for R package management. However, it is vital for reproducibility that packages are managed across PHI. An alternative method may be the recently released RStudio Package Manager.

Finally, in order to ensure that RAPs are sustainable, it is important that there is a minimum level of R and version control knowledge among analysts. The inclusion of code maturity, style and structure requirements in an analytical strategy will also help new or rotating staff to use, maintain and edit R code more easily.
References

Panko, R. 2016. What We Don't Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. Proc. 16th EuSpRIG Conf. 79-93.

