Version Control for Data Analysis

at Sheffield City Council

Laurie Platt

21 December, 2023

Table of contents

Preface **4**

1 Introduction **5**
1.1 Version control infrastructure 5
1.2 Risks and benefits 6
1.3 ToDo 6

2 Data protection **7**
2.1 Confidentiality risk 7
2.2 Information Management notes 8
2.3 Other ToDos 9
2.4 ToDo 9

3 Training **10**
3.1 ToDo 10

4 Code reviews **11**
4.1 ToDo 11

5 Anonymise **12**
5.1 ToDo 12

6 .gitignore **13**
6.1 ToDo 13

7 Repository template **14**
7.1 ToDo 14

8 Git hooks **15**
8.1 ToDo 15

9 Secrets **16**
9.1 ToDo 16

10 Fit for publishing checklist **17**
10.1 ToDo 17

11 GitHub roles 18
11.1 ToDo 18

12 Breach protocol 19
12.1 ToDo 19

References 20

Preface

This documentation is at an early stage of development.

This document includes good practice for using version control for data analysis at Sheffield City Council. In particular, it provides suggestions and requirements on how the data being analysed is handled by version control.

In addition to good practice, the guidance will also provide some pointers on getting started and links to other resources.

If you're reading this from the PDF version you can view the online version here:
scc-pi.github.io/version-control.

If you're reading this from the online version you can view the PDF version via the small Adobe Acrobat icon next to the title in index pane on the left, or by using this URL:
scc-pi.github.io/version-control/Version-Control-for-Data-Analysis.pdf.

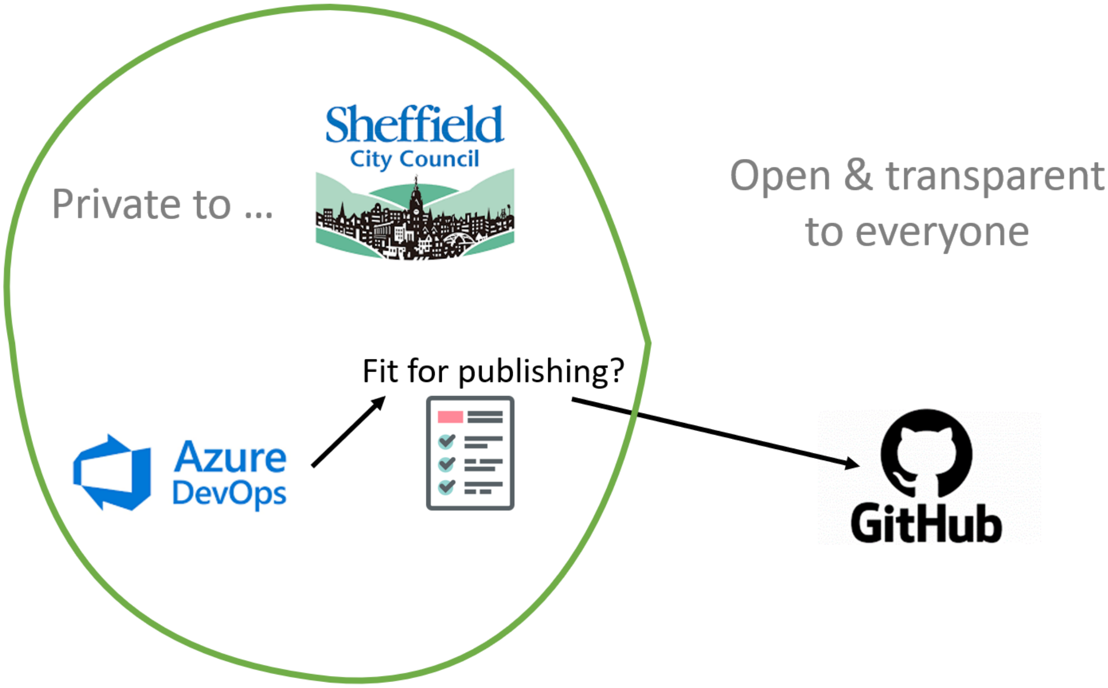
How this guidance is published is detailed in the repository README.md:
github.com/scc-pi/version-control#readme

The expectation is that this book, its repository, along with most repositories in scc-pi GitHub organisation, will move to Azure DevOps within the Council's secure Azure Tenant. Establishing the Council Azure DevOps for data analysis is currently on hold whilst a new data platform is established.

1 Introduction

A Show and Tell of version control was provided for data analysts at Sheffield City Council in August 2023. A recording and the slide deck is available from the Council's Data Network Sharepoint Site.

1.1 Version control infrastructure



We intend for Azure DevOps to be our default for hosting version control repositories. This will enable us to securely share code with data analysts across the Council.

Making code open and transparent to everyone, not just other Council Officers, is considered best practice. Making a repository publicly available will be done by publishing it on GitHub, but only if it passes a "fit for publishing" checklist.

Our adoption of Azure DevOps is currently on hold whilst we setup a new Data Platform. Until then, GitHub remains our current option. GitHub is subject to more data protection considerations, so until Azure DevOps is available version control by more Council data analysts is not being encouraged.

1.2 Risks and benefits

The risks of version control are mainly to do with exposing confidential data, whether it's commercially sensitive or PID (Personal Identifiable Data). Other risks are noted in the Fit for Publishing Checklist

To understand both the benefits of sharing code and how to manage the risks, we'll be leaning heavily on resources from: NHS RAP Community of Practice Government Analysis Function Guidance Hub

1.3 ToDo

- Some form of "Getting Started", so subsequent content makes sense. Links to other resources and training?
- Once this guidance is moved to Azure DevOps, include Show & Tell URL.
- Re-write chapter when Azure DevOps available.

2 Data protection

Using version control for data analysis requires data protection to be front and centre of your considerations, as it has to be with any form of data handling. That said, a lot of the risk mitigation will be undertaken as part of an unobtrusive version control workflow, once the workflow is established and well practised. Our procedures will also prompt for particular data protection considerations at different stages:

1. Project initiation and repository creation.

2. Code reviews with a colleague.

3. Fit for publishing checklist.

2.1 Confidentiality risk

The principal data protection risk is confidentiality. Confidentiality is defined as unauthorised disclosure of, or access to, personal data.

Version control for data analysis has three types of confidentiality risk. These relate to what a data analyst may inadvertently include in a **public** version control repository:

1. Data file (e.g. a spreadsheet) with personal data.

2. Some personal data in data analysis output or a script.

3. Secret (e.g. a password) in a script that compromises the security of, for example, a database containing personal data.

These risks are less relevant for SQL scripts than for R and Python scripts, and SQL scripts are more likely for new Council users of version control.

Our Azure DevOps repositories will not be public. Only our GitHub repositories will be public, so the GitHub repositories are the main risk.

2.2 Information Management notes

Once we move to Azure DevOps we may take advantage of the project management tools. However, for now we'll keep track of actions on this page.

Preliminary discussions with the Council's Information Management team have been held. Below are notes of (italicised) questions raised and (bulleted) actions agreed.

1. Are there retention schedules for scripts (and their versions)?
No. Current housekeeping is to periodically (maybe every 6 month) review the list of GitHub repos (repositories) and archive what's no longer in use.

- Formalise and document proposed housekeeping, including retention schedules.
- Extend periodic reviews of repos to include what should be archived, and what should be deleted from the archive.

2. When you publish your data analyses do you indicate anything about the script, its version and a plain English statement?
Each repository includes, as a minimum, a README.md file with a plain English statement. It will also often include further documentation about the data analysis pipeline. The current version and previous versions are an intrinsic aspect of GitHub and Azure DevOps.

- Include the README.md and documentation requirements in the version control housekeeping documentation.

3. When you publish to GitHub (in future) is it checked? attributed to SCC? and come with a disclaimer?

- Unit tests and code reviews are RAP (Reproducible Analytical Pipeline) practices that we're considering adopting.
- Our GitHub repos are held under the scc-pi GitHub organisation and are clearly attributable to SCC.
- Need to consider the different licenses that can be applied to published repos and whether this would cover the disclaimer.

Main action from meeting Information Management:

- Complete mini 2-page DPIA and make MW aware. It might sit under the full DPIA of the new Data Platform.

Other actions:

- Move GitHub repos to Azure DevOps.

- Draft a fit-to-publish (publically on GitHub) checklist.

- pre-commit checks (scripts) or “hooks” that screen for CSVs, PID, {secrets} (e.g. passwords) etc.

- Code review by another analyst will also lessen the risk of inadvertently including data.

- Publish guidance and tools for annonymising datasets for data analysis to encourage working with PID as only by exception.

- Protocols in case of a data breach from incorrect use of version control.

2.3 Other ToDos

- Replace readme with quarto doc and outline quarto book and github actions??
- Move version control content from pinsheff.

- Link to related unmoved pinsheff content.
- Note Version Control Show & Tell e.g. .gitignore /data.

2.4 ToDo

- What are the connotations of personal data being available to someone in the Council who doesn’t need access to it? For example, via Azure DevOps?

- Once this guidance is moved to Azure DevOps, include mini-DPIA PDF download URL.

3 Training

Version control is not especially difficult, but the risks will be greatest when someone is new to it. The low number of potential users of version control for data analysis means that bespoke Go Learn style training is not justified. Instead, there should be some training from an existing Council user of GitHub. This is in addition to personal responsibility for reading the Council guidance and undertaking specified, openly available, introductory training courses on Git and GitHub.

3.1 ToDo

4 Code reviews

Code reviews by a colleague are best practice that we should adopt. Another pair of eyes will reduce the risk of publishing personal data. Other benefits of code reviews include code health and knowledge sharing. We could look at defining some basic questions to cover as a part of a code review and include data protection considerations. The Government's Duck Book is a good place to start.

4.1 ToDo

5 Anonymise

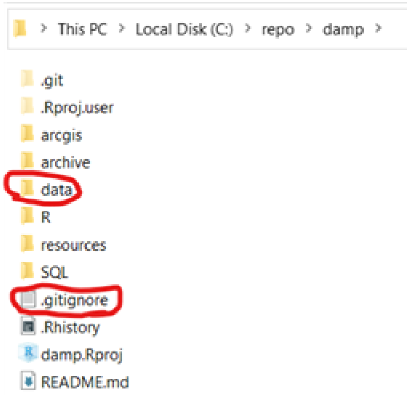
Where possible the data being analysed should have personal data removed or anonymised. We could better support this with written examples, functions, and packages, that all make it as convenient as possible to anonymise personal data.

The best way of removing the risk of leaking personal data is not to include it in the data analysis process in the first place. If you're extracting data from a database, don't select the personal data attributes unless it's necessary. If someone is providing a data extract for you, ask them to omit personal data unless it's necessary.

5.1 ToDo

6 .gitignore

For data files with personal data, the simplest way to ensure they're not published in a public repository is to store all data files in a `data` sub-folder in the project directory and then exclude the data sub-folder from the Git repository. This is done by specifying the data sub-folder in the `.gitignore` file.



› This PC › Local Disk (C:) › repo › damp ›

.git
.Rproj.user
arcgis
archive
data
R
resources
SQL
.gitignore
.Rhistory
damp.Rproj
README.md

6.1 ToDo

7 Repository template

We could create an SCC repository template to make consistent project folder structures and `.gitignore` exclusions more convenient.

7.1 ToDo

- Consider excluding initial data exploration and data engineering aspects from publishing.

8 Git hooks

Git hooks are scripts that can be set to run locally at specific points in your Git workflow, such as pre-commit. So, we can automate some checks for potential secret or personal data leaks by writing Git hook scripts to scan the contents of files in our version control repository.

8.1 ToDo

9 Secrets

The best way to avoid leaking secrets is to store them in local files. These can then be excluded from Git tracking with a `.gitignore` file. More likely and safer still, they will be stored within an IDE‘s folder structure (e.g. `.Renviron`) so that they are available across projects and are not exposed to specific project directories and version control.

9.1 ToDo

10 Fit for publishing checklist

We should adopt something similar to the NHS fit for publishing checklist, which includes data protection checks.

10.1 ToDo

11 GitHub roles

11.1 ToDo

- https://docs.github.com/en/organizations/managing-peoples-access-to-your-organization-with-roles/roles-in-an-organization
- https://docs.github.com/en/code-security/getting-started/best-practices-for-preventing-data-leaks-in-your-organization
- Retention schedules for repositories.

12 Breach protocol

12.1 ToDo

References