Checklist to validate repo before publishing to GitHub

For code quality, we recommend the analysts applying good engineering practices for building their code. Some of these materials can be found in our Data Science rap-community-of-practice and NHSD Software Engineering data-quality-framework. The below is only concerned with checks to ensure there is no sensitive information present, and that the code is safe to publish externally.

RED Not fit for publishing - requires immediate attention

AMBER Not fit for publishing - will require attention before external review

GREEN Fit for publishing

- **Repository to be published on GitHub:** <insert GitHub repo link here (if it exists)>
- **GitLab repository used:** <insert GitLab repo link here>

Checklist items	Int. Comments & Suggestions	Confirm suggestions have been implemented	Ext. Comments & Suggestions	RAG status
Name of checker				
Confirm code is fit for purpose. This could cover: 1. Chosen RAP level (e.g. Baseline/Silver/Gold) **if it's a RAP project.** 2. All outputs replicate what has been published or designed/developed for. Code adheres to standards for clarity, commenting, style, etc. **if it's a non-RAP project.**				
• Confirm code has been peer reviewed. • Confirm code is tested (e.g. unit testing, backtesting etc.) • Runs in a different environment e.g. runs on another machine (re-clone the repo, install the repo as a package, run the pipeline) and/or a different OS * [1]				
Identify clearly who owns the code and how others can use it				
• Obtain approval from the owner of the designed products or services in this repo to publish code. • Assign the person (or team) with responsibility for ongoing support and communications for the code.				

[1] Linux/Mac. Code should be put through a build test to ensure that it will run in another environment. E.g. promote code to test env, install and run. Or CI pipeline in gitlab using docker image – **This may not apply to all projects**

• Make sure the code does not include any unreleased policy or sensitive algorithm (e.g. fraud detection)				
Secret & credentials scanning[2]				
• Remove (**if any**): all passwords, IP addresses, SQL Server addresses, AWS secrets, and identification info. Anything that is intended for internal use only. • Remove Git history to prevent leakage of credentials or secrets in past commits (*in the fit-for-publishing process*). Downloading the repository's snapshot prevents this from occurring.				
Confirm no data stored in the Git repo				
• Ensure no Jupyter notebooks exist in the repo. • Keep any binaries, build artefacts/package files (e.g. wheel, egg) etc out of the repo. Double check the .gitignore file contains this items. • Logs should not be stored on the repo • Code comments in the script **that include any internal information** • Outputs (terminal, files, and database) or no credentials or PII data printed to screen				
Documentation				
• README should contain clear and key info of the repo (guide). • Select an appropriate licence and copyright for the repo. **Recommended Licence for the codebase is MIT and OGL 3.0 for documentation and should be Crown Copyright (see example).** • Add contact details (e.g. email) so users can request additional information or improvements. • Share the location of any available synthetic data or the metadata (if possible). • Link with the corresponding publication report				

[2] Credentials or secrets are essentially passwords that computers use for encrypted communication or access to services. For example, with many APIs (like the Google Maps API) you must supply a credential code to access the service. Often these codes look like long, strange combinations of letters and numbers (l79sDgH9s...). We must not share our passwords publicly, so you should not commit credentials and secrets.