**REPRODUCIBILITY AND RISK AUDIT**

**UTS | FTDI | MDSI | DSP – SPRING’19 - 13313491**

*“An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.”*

**OVERVIEW**

A code or an analysis that can be walkthrough by someone step by step attracts more trust. It makes the work reliable. Reproducibility doesn’t essentially correlate with correctness of the research; however, it helps gain trust of the reviewers and also makes it easy to identify any overseen defects. Publishing a reproducible work also facilitates others to easily adopt or extend the findings of the research to new work, enabling repeatability with minimal effort.

In the modern world, reproducibility of a digital work is not affected only by the code itself, but also by a number of other factors such as choice of programming language, operating system, version of software, packages and other dependencies. With increasing number of dependencies and problems for reproducibility, also are increasing number of innovative solutions such as evolving code style such as built-in unit tests, containerisation methods like dockers, version control systems, APIs, etc.

In this document I will be reviewing briefly about some of these techniques alongside a practical demonstration of how these techniques have helped improve the reproducibility and repeatability of a work that I carried out in the first semester for the subject Data Science for Innovation. The work can be reproduced by following the literate programming technique adopted in this link.

**DATA**

The analysis that has been re-done as part of this document was originally performed in Spring last year for Data Science and Innovation assignment ‘Quantified Self’. The assessment required a group of people working as a team to collect one set of data in a common repository on the cloud for use by every individual and another set of data to be collected individually be the members only for their individual use. The aim of the assessment was to perform analysis on the collected datasets, acquire insights from the dataset and produce a report. The code was not required to be shared at that time. The assessment was aimed at using the Data Science skills required in that subject to quantify about self.

At the time, I had used R to analyse the dataset, along with minimal use of visualization tools like Tableau, and Excel with plug-in to perform sentiment analysis. The data was sourced in two excel sheets was a mixture of quantitative and qualitative data.

**IDENTIFIED REPRODUCIBILITY ISSUES**

***The project space***

The project directory was a mess! Due to lack of proper organizing and version control, the project folder has files randomly named and spread across the directory, thus making it harder for even the author to reproduce or repeat the results after a period of time. This makes it harder for the code to be readily shared with others and limits reproducibility.

A screenshot of a cell phone

Description automatically generated

***Version control***

Lack of a version control like git, which prevents the changes that were made to the script over time or identify any problems or issues that occurred at the time of creating the scripts. Due to this, even when shared, the code could be subject to encountering issues which were originally solved by workarounds at that time.

A black and blue text

Description automatically generated

***Lack of documentation and comments***

The code has no reliable source of documentation or proper commenting, which makes it hard to interpret the choices of packages, codes, etc. Thus, when questioned or when in doubt about some of the choices made in the code, there could be no answers.

A screenshot of a cell phone

Description automatically generated

***Improper coding style***

No appropriate coding style was followed making it hard to interpret the code and thus debug. Also, the spaghetti code is confusing, reducing the interpretability even for the author of the code!

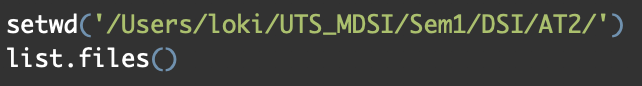
***Data confidentiality***

The data collected as a group had confidential details. Although the confidential details were masked, at the time, we had agreed only to the use of data within the member of the group and not share it outside.

**IDENTIFIED REPRODUCIBILITY RISKS**

***Use of absolute paths in the code***

Absolute paths were used in the code, which requires every user attempting to reproduce the results to alter these paths before they could verify the output of the analysis. Also, within the author’s computer environment, if the directory was to be renamed or moved, it would prevent the code from working in the future and require debugging.



***Lack of information of computational environment like package version, etc.***

There is no information available regarding the version of packages, OS environment the code was originally designed in, any path requirements etc. All these are having external dependencies that are subject to change or update overtime and could potentially cause the code to malfunction or stop functioning over a period of time.

***Lack of installation instructions like shell/CLI commands***

There is no guide on how-to set up the system to run the code.

***Lack of code sharing framework like github/bitbucket/gitlab***

The code was never shared in the first place nor does it have a cloud repository, thereby putting the code at risk of being lost, if the author’s computer were to encounter any issues.

***Lack of APIs that facilitate repeatability***

Provided the analysis from the code was found to be insightful and someone wanted to repeat the exercise with a new dataset, there is no API they could call to which would ease this effort. Instead everyone who wished to repeat the exercise performed using this code has to undergo each step manually to reap the benefits of the exercise. This step is a huge no in the world of computer programming that primarily aims to ease the operational efforts.

**IMPACT OF IDENTIFIED ISSUES AND RISKS**

***Issues***

* Project space does not facilitate easy sharing of code with others
* Lack of version control, unit tests, documentation and comments limits user from understanding the various issues encountered by the author at the time of development, and does not instruct on workarounds if the issues were to recur
* Coding style is confusing, thus making interpretation and debugging of the code difficult
* Data confidentiality restricts others from verifying the results of the analysis as performed by the author

***Risks***

* Absolute paths used in the code require alteration when executed on a new system
* Lack of computational environment information could lead to version/OS mismatch thus making the code unusable
* Lack of any set-up instructions makes it hard for others to have the project space setup for reproducing the results
* Lack of APIs to repeat the exercise makes it harder for the exercise to be replicated with new datasets

**REMEDIATION STRATEGIES**

Before diving into the remediation strategies that could help avoid the above identified risks and issues, let’s take some time to understand the need for Data Science projects to be reproducible and repeatable.

* Reproducibility is a key to gain trust of stakeholders. Often, people are satisfied only when they can verify an analysis by themselves.
* Often impressed by the one-off analysis work of a Data Scientist, there is a request for the analysis to be repeated on new datasets
* Data Science projects that begin as one-off could quickly turn into production grade requirements. Having a well-documented, structured and containerised code helps in getting the code from one-off to production-ready quickly.
* Data Science being hugely benefitting on knowledge-sharing platforms requires proper documentation to be able to share with fellow data scientists who may benefit from the work carried out and also help enhance the code overtime.
* Above all, a well modularized code allows re-usable modules/units to be quickly picked and plugged-in into new projects, thus making the life of Data Scientists so much easier!

**REMEDIENG IDENTIFIED REPRODUCIBILITY ISSUES**

The code for analysis was originally written in R. However, as R is not my primary programming language, for this assignment I have chosen python as the language of choice, primarily due to the fact that I am much more proficient in python and object-oriented programming style, in comparison to the functional programming style which R excels in. The choice of programming language in this case has no relation to the reproducibility.

***The project space, Version control***

Use of version control via git solves both of the issues identified here. Version control eliminates the need for having multiple versions of the codebase as separate files. All the changes can be made to a single file, referred to whenever necessary and also switched to if there is a requirement at any point of time. Version control also allows tracking of issues encountered, and workarounds added as features to the codebase over a timeline. Version control over cloud also has the advantage of facilitating multiple programmers to collaborate simultaneously over a single code.

Further, there are several GUIs that are available for download that helps visualize the changes to the code, compare them side-by-side easily.

***Lack of documentation and comments***

Literate programming in iPython notebook, an interactive notebook that let’s code, informative comments and interactive visualizations all live together simultaneously within a single file is used to capture the re-formatted code along with the comments on what the code does, how to use it and interpret the results.

***Improper coding style***

Object-oriented programming style is adopted to rewrite the code. OOP concepts come with its own set of benefits in programming and is the most commonly adopted programming style for python. Modularization of code will also be carried out to facilitate calling functions from both the iPython notebook and other python files that will work as APIs with command line interface.

***Data confidentiality***

The data with names masked will be made available. Also, instructions to create similar toy-dataset or self-recreate the dataset over a period of time will be provided.

**REMEDIENG IDENTIFIED REPRODUCIBILITY RISKS**

***Use of absolute paths in the code***

Replace all absolute paths with use relative paths and enhance with python packages os, sys

***Lack of information of computational environment like package version, etc.***

Prepare a requirement file by freezing the pip package requirements and making this available along with the code, and CLI installation instructions thereby facilitating any user to quickly set up their python environment with the same package versions as the author.

If possible, create a docker container to containerize the whole exercise.

***Lack of installation instructions like shell/CLI commands***

CLI instructions will be added in the cloud code repository as explained in the above step.

***Lack of code sharing framework like github/bitbucket/gitlab***

The codebase along with dataset, requirements file, and readme file for instructions will be made available in GitHub.

***Lack of APIs that facilitate repeatability***

Modularized python files that can be called from terminal will be added to the repo and can be used to quickly generate the outputs.