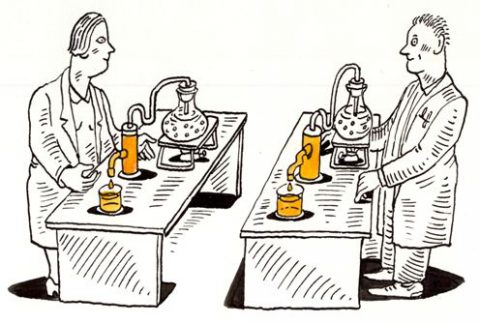
Reproducibility and Risk audit

Zarmina Muhammad

31 October 2019



<https://blog.f1000.com/2014/04/04/reproducibility-tweetchat-recap/>

# Why is Reproducibility Important?

In data science world, Data & Analytics (D&A) is the key to unlock the rich by effectively interrogating and understanding data. Data doesn’t only bring greater assurance to organizations and markets, but also brings attention to performance, risk and regulation issues. That is why reproducibility, specifically computational reproducibility has become an imperative constraint of every data science task.

**“The most important is the mindset, when starting, that the product will be reproducible.”**

***Keith Baggerly***

This report addresses the reproducibility and risk factors associated with one of my previous assignments from DSI on mood analysis (<http://rpubs.com/ZarminaMuhammad/544502>).

# Overview:

The report will determine **“How mood is affected by the digital screen time and is there could be any relationship between the two parameters?”**

The overall intent of this assignment carried out by our team “AIrborne Analytics”, was to collect the data about ourselves in context of mood, over a four-week period time and to determine the relationship between mood and screen time. For this analysis, we added different other parameters down the track like academic events (assignment deadlines) to determine its impact of these parameters on the mood of a person. The analysis was performed using;

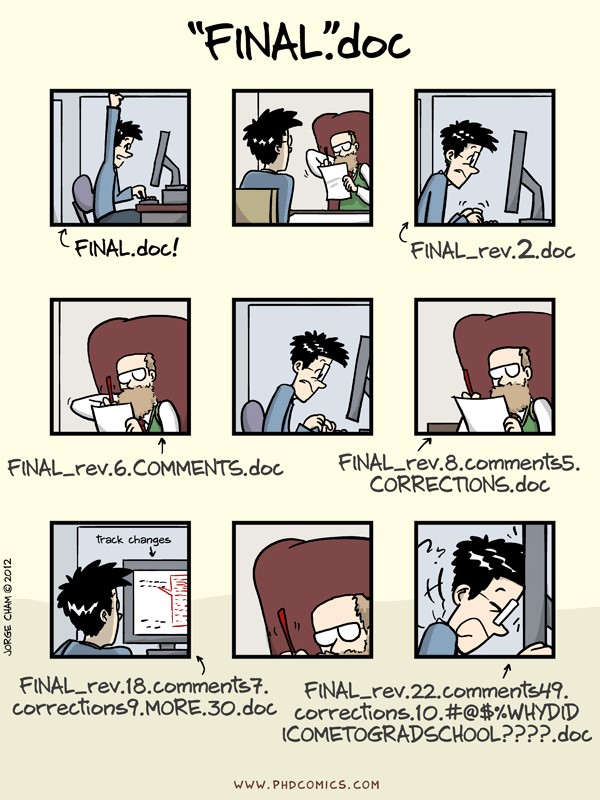
* Excel: For dataset joining and missing values interpretation
* Tableau: For visualization
* R: For personal data analysis (google timeline)
* Microsoft Word: For report writing

# The problems/issues in the report:

Looking back and scanning through my previous submissions with an objective to correct the incorrect, I realized that it was a painful task in identifying how many problems were associated in reproducing my very own work which I had done no longer than 3 months ago. Consequently, I also realized that how difficult it would be for someone else reproducing my work.

The problems I found with my work were as follows:

* Are the figures reproducible from the code and data?
* There is no record of the series of steps taken that shows the generation of output. I generated a huge report on my analysis, but had no record of my performance for the strategies or steps I had taken to reach to conclusion.
* Data files, analysis, visualizations all are not part of the report.
* After modifications in one of the files, it was still unclear which version corresponded exactly to the reported results, as it involved series of modifications therefore leading to multiple versions of one document i.e. Microsoft word documents with confusing naming conventions such as final, modified, submission copy etc.
* I found it so difficult finding the actual R-code I had written for my personal data analysis.
* No backlog of the analysis performed, was present within the report so that to distinguish the visualizations based on the data, from where it was generated or on the type of analysis, for another reader.
* If any slight changes occurred within the data or the visualization, then it became very time consuming as the steps involved in naming figures, outputs in the form of results and reports must be regenerated, as the reports were not automatically updated.
* The reports were not linked to the data sets and the visualizations tools. That’s why slight amendments in the data became too hectic. (I realized how painful it was, while converting my report to an RMD file from a word file. I felt that my situation same as the one in the figure below.



<http://phdcomics.com/comics/archive.php?comicid=1531>

Keeping in mind all these problems that I faced regenerating my own document, I realized how painful it is going to be for someone else in terms of regenerating the same concept and that how important REPRODUBILITY is in a data science task. Steps towards improving and reducing the above-mentioned risks and reproducibility issue from Data science practice (DSP), I have learned how to improve reproducibility of a project. This mayinclude writing a report that can generate results, where:

* The report is automated via code
* Data is attached to that well-documented code
* History of any changes is preserved
* The final report should be self-sufficient and reproducible with a single command

# Tools to enhance reproducibility:

**Correct the Incorrect!**

## 1. Version control – what and when did you do?

**Version control** is a system that records changes to a file or set of files over time so that you can recall specific versions later. Git and GitHub – version control system

* Git is a command line tool
* GitHub.com is a web-based storage for your project repositories

To keep track of all the steps in my project version control is the best solution. GitHub stores my project in its own repository either public or private depending on my preference (where I can access/share my code with anyone, anywhere and I am not confined to using my personal machine rather it’s available in the cloud).

Another issue was to keep track of series of steps during the entire project. This could be solved by using Git/ Git Bash, where I could keep track on my process as well as go back to any step and make changes. The branching in Git allow me to try different things with my code and then merge with the code, otherwise not affecting the master.

In case of collaborative work, such as the project being discussed, each user has to share his/ her own data, so using Git, the merging form multiple people was possible, that would keep track of all the members in a team regarding their sharing datasets and making amendments in their own datasets without affecting the rest of the data. The collaborative ability of Git could also be used if the report/code is being written by multiple people.

Git not only keep track of the entire process but also if something must be removed or changed with in the code/report, Git easily undo changes, and then re-apply all the changes being made after that.

## 2. SQL: how to join the datasets

Combining different data sets from multiple users was a tedious job doing in excel. The was done based on unique number assigned to each user and then combining all consolidated datasets into one based on timestamps.

This could be easily achieved through SQL with basic **select** and **joins** and some other functions. So, the entire hectic jobs would be reproduced with few lines of code. The benefits of using SQL in terms of reproducibility included;

* An hourly job of joining datasets is done by few lines of code
* The dataset remains within the report and could be easily accessed by someone else.
* RMarkdown could run SQL query so the data would remain within the report.
* If certain changes were to be made to the dataset i.e adding/removing a variable, or adding another dataset, then this may not impact the entire analysis and the report structure. Therefore, accommodating new data to the analysis won’t be a nightmare anymore.
* Since Git is keeping track of all changes, so looking back on progress is not going to be a problem any longer.

## 3. Automate everything

**“If you torture the data enough, nature will always confess.”** ***Ronald Harry Coase***

### R/ R-Studio

Excel, PowerBI, Tableau and others are widely used tools to visualize data. However, R offers a plenty of possibilities and benefit in those regards. R is a free/open source programming language that runs on Windows/ Mac/Linux. R is powerful in terms of having a very large collection of actively developing packages and has excellent graphics & report-creating capabilities.

In this analysis I have used tableau for visualizations, though I could have used R (such as ggplot2, plotly and leaflet). In terms of reproducibility, the data being already processed in R (SQL query), the benefit of using R would include;

* No loading of data sets would be required for analysis
* The visualization from the analysis would remain in the report, and no uploading of external images would be required (eliminating the chances of adding wrong image).
* Less time consuming
* Automation: Report generation required a single command.
* A single tool for end to end production (can accommodate SQL/ python query through specialized packages)

### Python

Like R, python is also a very powerful language for computing data science tasks and has its own strengths like machine learning via scikit-learn, developer support and web scraping and working with APIs.

Using **reticulate package** a python environment can easily be created in R, so if a task requires a task to be efficiently done, it is doable in R and that task can be a part of the final report as well (no import required for any graph or table, everything lies within the R environment)

## 4. Literate Programming

From DSP, one of the greatest benefits is to efficiently write a report. A program is written as an explanation of the program logic in a natural language (English) along with chunks of code The benefits of creating reports using these tools include;

* A report contains a stream of text and code chunks
* Each code chunk loads data, computes results, showing figures
* Each text chunk explains how the code chunks work
* The resulting report is human- and machine readable

Examples of literate Programming are

* RMarkdown (creates output in HTML/pdf/ word)
* Notebooks

### knitR (RMarkdown)

If I had previously done my entire analysis just the way I mentioned above, I wouldn’t have had gone through the pain of reproducing my own report utilizing many hours. Using RMarkdown, report generation becomes an easy task. If certain changes are to be made in the analysis, it no more relies on performing all tasks manually, rather, make changes to the code or the text and then knit it, a report will be generated with simple commands.

**How markdowns work:**

knitR - a package for dynamic report generation written in R Markdown. (Supports RMarkdown, LaTeX, MathJax. PDF, HTML, DOCX output)

**Properties**

* Code chunks separated by ```. Inline code allowed
* Graphics, code generated and external images
* Tables, code generated and in RMarkdown format
* Caching long code chunks
* Code chunks and results output fully customizable

### Reproducibility for other languages (Notebooks)

Just like RMarkdown, in a notebook a program is written in multiple snippets including markdown where natural language-English is used for explanation of the program (code, insights, outcomes and conclusion), interspersed with snippets of code. A fully documented notebook includes procedures, data, calculations and findings. IPython and Jupyter Notebooks are the two types of notebooks.

**Properties**

* Combine text, equations, code, and graphics
* Markdown support
* Multiple languages support (>40)
* much easier to reproduce results and findings because the reader has everything, they need to understand what you did in your project.

## 5. Reproducibility for the whole projects

**Docker - an envelope (or container): lightweight virtual machine.**  Containers are a way to package software in a format that can run isolated on a shared operating system. It encapsulate operating system components, scripts, code, and data into a single package that can be shared with others, providing operating-system-level virtualization by abstracting the “user space”. The idea of a container is that, it is an isolated environment in which we can set up the dependencies that we need in order to perform a task.

**“Operators”** use Docker to run and manage apps side-by-side in isolated containers to get better compute density.

**“Enterprises”** use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux, Windows Server, and Linux-on-mainframe apps.

**“Data industry”** use this for performing ETL work, monitoring data quality, standing up APIs, or hosting interactive web applications.

The overall goal is to provide isolation between instances with a lightweight footprint.

### Containers in a Data science Environment:

Using Docker is going to take more work for a data scientist, versus a local deployment. However, there are several benefits to Docker containers:

* **OS-independent, portable application images:** It eliminates the “works on my machine” problems when collaborating on code with co-workers.
* **Preserves all application dependencies:** It depends on libraries and settings to make the software work, thus making it efficient, lightweight, self-contained systems that guarantees that the software will always run the same, regardless of where it’s deployed.
* **Easy to distribute and reproducibility:** analysis can be delivered easily as a container and other data scientists can rerun the same work.
* **Build once run anywhere concept:** Docker has made it much easier for anyone to take advantage of containers in order to quickly build and test portable applications.

### Use of containerisation as a strategy for reducing certain reproducibility risks

* **incorporating the markdown into a software container:** In my analysis I have used literate programming- RMarkdown (that combines narratives with code), so incorporating the markdown into a software container so that others can execute it without needing to install specific software dependencies would reduce reproducibility risk.
* **Incorporating workflow management system:** In case of using multiple tools such as, SQL, R and Python all together in an RMarkdown environment, the container might also include a workflow management system to ease the process of integrating all these multiple tools and incorporating best practices for the analysis. This container could be packaged within a virtual machine or cloud-computing environment to ensure that it can be executed consistently.
* **Reduced risk of Technical debts:** Risk of technical debts in R-code is reduced, as a container is not dependent on the R-Studio version installed on local machines, but it runs on its own set if libraries and settings that the software is built on (self-contained systems). It will always run the same regardless of where it is deployed. This also means that can use older versions of a package (Suppose R-3.3) for a specific task, while still keeping the package on your machine up-to-date (R-3.5).

Although, still under active development, such services are signs of the future for computationally reproducible science in this data driven world.

# Reflections and Conclusion

Reproducibility refers to a task/project that is replicable, reliable, robust and repeatable. It requires extreme efforts and considerable amount of time to perform an analysis that is replicable and reliable.

Looking back to my previous analysis, and converting it into a literate programming format, was painful and a time-consuming job. I realized the importance of reproducibility and the associated risks with such tasks, after taking long hours replicating my own work from not long but 3 months back and how terrible it would be for someone else repeating this task, or even for me making some amendments down the line 3-4 years from now.

However, Data science practice, helped me learn tools and techniques that when applied such data science task result in efficient reproducibility, requiring only standard procedure to get to the desired output.

Applying all these techniques to my task would result in

* a completely automated report (literate programming) that
* includes code and data (along with procedure and insights),
* generated on proper guidelines, and
* reproducible by someone else generating the same output.

**Improving reproducibility:** During the re-analysis of my previous work and DSP, I learned how to improve reproducibility and eliminate/ minimize risks associated to my work in the form of,

* keeping in mind the experimental design, analysis and product/outcomes from the analysis.
* **Keeping history of changes** via code versioning and sharing: using **Github** a remote repository containing my files are stored and can accessed and shared with anyone form anywhere whereas, **Git** keeps record of all the changes that has occurred during the entire project and can trace back on specific steps when required.
* Using **literate programming (self-documenting code)**, such as Markdown or notebooks to have data, analysis, visualizations, results, findings and interpretations along with proper documentation within one file (report).
* **using containers**, the overall project becomes independent of local machine and can be accessed from anywhere, could be shared and where it could be rerun by someone else as well. Containers make deployment of a project easy, anywhere anytime independent of the local machine and is super fast.

# Appendix:

* Rewritten report using RMarkdown: <http://rpubs.com/ZarminaMuhammad/544502>
* GitHub Repository: <https://github.com/ZarminaMuhammad/DSP-reproducibility-and-Risk-Audit>
* DSP course guide: <https://datasciencepractice.study/>