**Abstract**

Due to the recent advent of open-source and collaborative paradigm, sharing and collaborating in IT engineering sectors have been blooming, which consequentially contribute to the breakthrough of the overall productivity and efficiency in industry. This recent trend also has influenced on academic research schemes. That is, the reproducibility is highly emphasized in addition to the result of the research. Bringing in the notion of reproducibility to academic research not only helps to publicize his or her work easily but also provides a means to do a sanity check on his or her work. In this paper, I present five fundamental tools that make reproducible workflow easy and efficient.

**1. Introduction**

Reproducible research provides four key benefits namely, transparency, openness, efficiency and productivity. 1) Transparency provides the credibility on your research and its result. If the readers can follow every step in your research, they also can do a sanity check and trust your work. 2) Openness indicates the accessibility of your work. Once your work is publicized, other research can make use of your work to add more values or derive other research questions. If your work is reproducible, it is helpful for others to reproduce and make use of your finding. 3) Efficiency is increased because other people do not go through the exact same analysis from the scratch. If your research is reproducible, they can immediately use your work. 4) Productivity, as a consequence from previously mentioned benefits, can be approached.

To successfully maximize the benefits, you need to have right tools for reproducible workflow in your toolbox. This paper presents four basic tools: Makefile, Git, GitHub, pandoc and Markdown. To effectively introduce these tools, I organize this paper in a way that the readers can obtain the general big picture of each tool and its role in reproducible workflow by relating to user scenario in practice.

**1. 1 Makefile**

A Makefile is a file containing a set of directives used with the make build automation tool. The *makefile* directs *make* on how to compile and link a program. Using [C](https://en.wikipedia.org/wiki/C_(programming_language))/[C++](https://en.wikipedia.org/wiki/C%2B%2B) as an example, when a C/C++ source file is changed, it must be recompiled. If a header file has changed, each C/C++ source file that includes the header file must be recompiled to be safe. Each compilation produces an [object file](https://en.wikipedia.org/wiki/Object_file) corresponding to the source file. Finally, if any source file has been recompiled, all the object files, whether newly made or saved from previous compilations, must be linked together to produce the new executable program [1].These instructions with their dependencies are specified in a *makefile*. If none of the files that are prerequisites have been changed since the last time the program was compiled, no actions take place. For large software projects, using Makefiles can substantially reduce build times if only a few source files have changed [2].

Suppose you are analyzing users’ behavioral patterns in a certain social network service. You are planning to do an exploratory analysis, preprocess data, implement models and write a report. So, you want to first create several, separate folders and files in order to decouple so that files-management would be easier and neater. Also, since your files depend on each other, you should deal with dependency. Furthermore, some scripts take long time to compile, and therefore, you want them to be recompiled only when they are changed. Lastly, you want your readers to easily reproduce this step through one single command that automatically make your project structure all at once. This is a right time to use “Make” and “Makefile”.

**1.2 Git**

Git is a [version control system](https://en.wikipedia.org/wiki/Version_control_system) that is used for [software development](https://en.wikipedia.org/wiki/Software_development). [3] For example, during the course of your project, you want to keep track of your work by saving your code files on a regular basis. One of the benefits of using Git is that you can always add new changes and undo changes so that even if your files corrupted or you made mistakes such as accidentally deleting files, you can always go back to clean states. Also, you can make branches to work on features separately and then merge branches together later. This helps you divide and conquer piece by piece. Furthermore, since Git keeps track of your changes, you can create different versions. Suppose you want to write a script that calculates the retention rate of user. You can write it in several different ways. Then, you can create branches dedicated to different versions of the script. Later, you can choose one of them and merge it into your master branch. Not only that, you can easily see the commit messages you wrote for your commits.

**1.3 GitHub**

**GitHub** is a web-based [Git](https://en.wikipedia.org/wiki/Git_(software)) [repository](https://en.wikipedia.org/wiki/Repository_(version_control)) [hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of [Git](https://en.wikipedia.org/wiki/Git_(software)) as well as adding its own features. It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project. [4] For example, now you want to store your codes to cloud storage so that you can have accessible to the files anywhere. Then, you can push your codes tracked by Git to GitHub. Once you upload your project, you do not need to worry about losing your laptop. Furthermore, suppose you are working on a research project with other members. If you use GitHub, you do not need to copy and paste your codes to USB to share. By uploading your project to GitHub, you can effectively publicize your codes to everyone unless you make it private. Thus, the GitHub is the hub of open-source projects and collaboration. You can not only store your codes to it but also download someone else’s works and make use of them easily.

**1.4 Pandoc**

Pandoc is a [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) [document converter](https://en.wikipedia.org/wiki/Document_converter). [5] For example, you write a report in a markdown, and also want to convert to other file formats such as pdf, html or plain text. You can easily convert any file formats to different file formats using Pandoc. This is really convenient because you can convert your report into a variety of formats without manually copying and pasting it. You can also specify this command in a makefile so that when compiling your files, it will automatically generate a report in different formats.

**1.5 Markdown**

**Markdown** is a [lightweight markup language](https://en.wikipedia.org/wiki/Lightweight_markup_language) with plain text formatting syntax designed so that it can be converted to [HTML](https://en.wikipedia.org/wiki/HTML) and many other formats using a tool by the same name. [6] Your effort on writing a good analysis and report boils down to writing a well- formatted and readable result. Markdown provides an easy solution to this task. Also, you can easily convert it to HTML format so that you can effectively put it online without manually changing it into an html file.

**2. Discussion**

This section discusses my personal experience on dealing with these tools introduced above. Therefore, this content is meant to be self-reflective.

**2.1 Resources I used**

Because these tools are well known and widely used tools, there are a lot of tutorials and documentations on Internet. I first read Wikipedia and introductory articles in official websites to understand the basic core concepts about these tools. Also, there are a lot of easy-to-follow tutorials that provide a step-by-step procedure from installation to basic commands.

**2.2 Easy parts**

I have already been exposed to Makefile, Git and GitHub from my computer science courses. Thus, using these tools is very natural and easy for me.

**2.3 Challenging parts**

When I first encountered these concepts such as branching, merging and pushing, it took me a while to digest. One of things that help me understand during that time was to use visualizations to understand the workflow graphically. Also, since I did not often use makefiles, I had to go over some tutorials to get used to both syntax and structure.

**2.4 Time consuming parts**

Installation was very straightforward and easy, and some of them are pre-installed in OSX. Time consuming part was to make a well-formatted markdown file. It involved a trial and error process. But overall, everything was quite intuitive and thanks to the tutorials, I could seamlessly go through all parts.

**2.5 How much time I spent**

I spent eight hours in total. To decompose the time I spent, 1) 1 hour to install all the relevant programs. 2) 2 hours to go over concepts. 3) 2 hours to have a hands-on experience. 4) 3 hours to write a report.

**3. Conclusion**

In this paper, I provide five basic tools for reproducible workflow for your project or research. There tools cover from making directories and files to managing your codes through Git and collaborating with others using GitHub. Once you get used to these tools, you can effectively take advantage of them for in your daily reproducible workflow.

[1] ["Introduction to GNU Makefiles"](https://www.gnu.org/software/make/manual/make.html#Introduction). https://www.gnu.org/software/make/manual/make.html#Introduction

[2] https://en.wikipedia.org/wiki/Makefile

[3] *Scopatz, Anthony; Huff, Kathryn D. (2015).*[*Effective Computation in Physics*](https://books.google.de/books?id=DYoNCgAAQBAJ&pg=PA351)*. O'Reilly Media, Inc. p. 351.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*9781491901595*](https://en.wikipedia.org/wiki/Special:BookSources/9781491901595)*. Retrieved 20 April 2016.*

[4] Williams, Alex (9 July 2012). ["GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz"](http://techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/). Tech Crunch. Andreessen Horowitz is investing an eye-popping $100 million into GitHub

[5] ["Pandoc Converts All Your (Text) Documents"](http://chronicle.com/blogs/profhacker/pandoc-converts-all-your-text-documents/38700). *The Chronicle of Higher Education Blogs: ProfHacker*. Retrieved 2014-06-27.

[6] ["Markdown"](http://daringfireball.net/projects/markdown/). 2013-12-04, http://daringfireball.net/projects/markdown/