The importance and implementation of rich documentation in bioinformatics software projects

Omar Rosado Ramírez

May 12, 2019

Abstract

Large software projects, specifically computational biology research projects, require a rich documentation scheme that can guide generally any user through the process of downloading, installing and running the project. Kevlar, a genomic sequencing framework is an example of a large project that has a documentation that falls behind it's size. Upon downloading Kevlar, the installation process takes the user to multiple pages that only lead users to copy paste the first command they find to see if it works. Installation of software should be basic and smooth, or users might seek software elsewhere(proprietary, not free software with an easy install). Considering the nature of the Kevlar project, it highly needed a user friendly documentation that let users to a successful, first try installation that would result in users actually using the software. This paper focuses on successfully setting up and running Kevlar on multiple platforms while also creating a documentation that better suits the potential user traffic of an open source project of its nature and size.

You can download the full Kevlar with the new documentation here: `https://github.com/afrotonder/kevlar`

Introduction

Genome sequencing and sequence assembly are a few of the most hot topics in computational biology, bioinformatics and computer science overall. Due to the management of large genomic data, projects of this nature tend to get big fast and as a consequence require much maintenance and a rich documentation. The general problem in genomic data handling is the amount of computational power required to analyze large amounts of data with the least amount of errors, which require large and complex algorithms.

Recently, many open source genome management tools have been released for the public to use as research software. As mentioned, the documentation of these research software requires for them to be insightful, productive, and accessible considering the user traffic it might attract and the possible longevity of the project. Open source projects are also made to be used by others who do not necessarily have the same experience as the author(s), which is another reason a good documentation is key in making good research software, or any software to be exact. As stated by Dr. Winston Royce in his paper *Managing the Development of Large Software Software Systems*: [1], "Without good doc-

umentation the software must be operated by those who built it"(Royce, W. 1970, p.5).

Kevlar [2], a newly released open source genome sequencing framework is an example of a big, successful project with a documentation that seems 'default' [1]. An online documentation is provided, although it is scattered between a website and various githubs, and it takes a while to get the software running. A README is available with the project bundle and on its official github, but they both lead to the same array of links and github accounts. Once Kevlar is set, it becomes the ultimate framework for genomic data handling. Along with an integrated CLI with subcommands and an automated Python-driven workflow, Kevlar is capable of much more than its documentation structure show.

Re-documenting Kevlar will probably increase its productivity by enabling users to easily install and run the software. Including a full documentation will also likely enable users to run Kevlar without leaving the development environment, something not possible before this paper.

Methods

Open source projects such as Kevlar often require a work environment with all its dependencies available. After setting up an environment and installing all dependencies, software should normally run. If all dependencies are not listed together, something is bound to get lost between the lines. Documentation of each step is required to maintain the project integrity on every installation. To better structure a project, work done must be documented in two important places: a README and a Manual. README's are plaintext files that hold information pertinent to the software such as **a description of the software**, **downloading the software**, **the steps necessary to get the project or software running**, etc. Manuals or 'Man pages' are another form of documentation found in UNIX environments that act as a 'quick help' and contain a more mechanical explanation of the software it represents, detailing each command if available and providing running information along with examples. Although it is not a requisite to have both methods of documentation, having both will greatly increase the ease of use of any software project.

Implementing Documentation Types

After all software requirements were gathered, documented and installed, testing could commence. Kevlar has the ability to be run in a step-by-step manner to better gather data from each part of the project and also in a workflow manner that bundles all of Kevlar's functionalities and runs them as one program. The step-by-step manner is run with the Kevlar CLI, a command based workflow that lets you call each of Kevlar's functionalities such as counting kmers, aligning contigs and calculating a likelihood score for unique reads (see [2], p.8-10). To run the Kevlar CLI, the user had to go to a separate URL to find all of the available commands with examples. Although Kevlar provides a 'help' command that displays a list of these CLI commands, they are displayed without flags or possible parameters. To solve this, a UNIX Man page entry was created using the online CLI documentation mentioned above. Man pages have

the advantage of being available offline, in-terminal and are completely editable if anything should be added, making Kevlar more accessible users.

The workflow manner is handled differently and requires an extra software dependency called Snakemake [3]. Setting up Snakemake is a bit line consuming, so these steps we're not added to the Man page. Instead a new section was created in the project README specifically for the Snakemake workflow along with each step and command available [4], plus another dependency [5] found along the way .

After documenting both ways of running Kevlar, the project README seemed to be taking a new direction. Kevlar's old README was a short, linked description of what the software did, where to get it, and where the documentation was. Essentially, the README was pretty 'default'. The next step was to revamp the README into a new, fully informative source that helps any user install, run and enjoy Kevlar. The new README was divided into 7 major parts: Description of project, Setup Virtual environment, Download Kevlar, Download Genomic Test data, Running Kevlar, Setup Development Environment, and Miscellaneous Information. Each part was derived from all different parts and links of Kevlar's documentation and only excludes each Kevlar CLI command, which would have made the README too long.

Testing the software with new documentation

The restructuring of Kevlar's README has improved the projects productivity. The next step was testing Kevlar with the new documentation on different platforms or operating systems. Kevlar and it's dependencies are pure Python and depend on a UNIX-like environment to run in. Microsoft Windows is one of the most used operating systems, but it has no easy way of running Kevlar unless you install a virtual machine, which is why this operating system was discarded for testing. UNIX-like operating systems chosen to test we're the two most popular to users: macOS and Ubuntu.

Ubuntu: Test passed

Following Kevlar's new README, Ubuntu received a smooth installation of all dependencies and the framework itself. Privilege errors may occur, but that depends on how and where the virtual environment was configured.

macOS: Test failed

With Kevlar's old README, installing the software led to multiple errors. On creating a virtual environment and installing all dependencies, snakemake was the only software not installing. It was apparently due to a dependency of snakemake, which depends on a Python module called cython that has a line of code considered deprecated in Python3.7. It is a recent and common error seen in several different Python projects like [6] and [7], and is solved by installing Python3.6 as seen in this [8] other Python project with a similar error. This error has not been officially corrected, so Python3.6 will be needed for Kevlar on macOS in the time being.

After eliminating the error mentioned, the Python package manager pip started giving errors due to version incompatibility. The pip package installed

was done so with Python2, which is currently at the end of it's lifetime and developers are encouraging abandoning it for the more stable Python3. Due to this, pip was completely removed from the Kevlar virtual environment and reinstalled with Python3 for future use. After these steps were documented and

Documenting these steps provides a clear path for those looking to install and use Kevlar on any UNIX-like platform. With Kevlar's newly documented README, re-trying the full installation resulted in error-free success.

Results

Following Kevlar's documentation proved to be challenging. Dispersed between two different websites and 3 different githubs, the setup took at least one hour. Running Kevlar took about twenty minutes due to the running commands and specifications being in two pages of a same documentation site and a github account. The project is excellently built, but was probably built by a group of developers and is being used by the same group of developers, which do not necessarily require a 'by-the-hand' walkthrough on how to install and run the software.

Running Kevlar in two different operating systems led to different results, adding additional steps to the installation process. The operating system ubuntu displayed an easy install while macOS backfired. Python version conflicts arose on macOS and some modules had to be replaced manually, causing the process of installation to become longer. Without testing and documentation on different platforms, software projects are limited to the environment it was created in, which isn't always similar between users with different versions of different operating systems on different computers.

Implementing a new documentation scheme to Kevlar improves it's productivity by making it simple to install. Instead of having to visit five different websites to setup Kevlar, now users only need to have the project locally and all the information needed is available. The new README provides a step by step process on setting up a virtual environment, installing all dependencies and running the Kevlar Snakemake Workflow. A UNIX Manual also included in the project improves Kevlar's user experience by enabling them to access each CLI command in-terminal and with examples.

Conclusions

Open Source genome sequencing research projects are becoming more common every day, which opens a large field for many users to contribute in. When projects become sufficiently large and require maintenance, the original developers might not be contributors anymore. This presents a problem for these types of projects if no well structured documentation scheme is implemented. Users might not all be experts on the subject, so it is imperative to have a well defined README that contains at least the following parts in detail: **description**, **how to download**, **how to install**, **how to use** and **examples**. Including specific and important steps such as software version differences and possible platform dependencies can make installing and using any software less of a pain for experienced and inexperienced users alike.

Future Work

Considering Python2.7's deprecation at the end of this year(2019), Kevlar should be adapted to Python3 to avoid future problems. It is imperative to document this change, for the contrary would imply users running the software to find many errors that are not easy to dig through.

Acknowledgements

I would like to thank Dr. Humberto Ortiz-Zuazaga for his constant support and mentorship. I would love to specially thank him for accepting me into his lab a few weeks late into the semester after discovering I was missing just 1 credit to graduate. A big thank you to Dr. Daniel Santange for indirectly giving me the chance to contribute to an open source project. Also, I would like to thank the students at the megaprobe who helped me shape my project during the semester.

References

[1] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[2] Daniel S. Standage, C. Titus Brown, and Fereydoun Hormozdiari. Kevlar: a mapping-free framework for accurate discovery of de novo variants. *bioRxiv*, 2019.

[3] Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 08 2012.

[4] GoogleAPIs. kevlar-workflow. `https://github.com/kevlar-dev/kevlar/tree/master/kevlar/workflows/mark-I`, 2019.

[5] Heng Li. Burrow-wheeler aligner(bwa). `https://github.com/lh3/bwa`, 2019.

[6] Sheffield Machine Learning Software. Gpy. `https://github.com/SheffieldML/GPy/issues/649`, 2018.

[7] scikit-learn contrib. py-earth. `https://github.com/scikit-learn-contrib/py-earth/issues/191`, 2018.

[8] GoogleAPIs. google-cloud-python. `https://github.com/googleapis/google-cloud-python/issues/3884`, 2017.