Ten Simple Rules for taking advantage of GitHub in bioinformatics

Yasset Perez-Riverol, Rui Wang, Laurent Gatto, Timo Sachsenberg, Juan Antonio Vizcaíno1,\*

1 European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD, UK.

\* Corresponding author: Dr. Juan Antonio Vizcaíno

European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD,UK. Tel: +44 1223 492 610; Fax: +44 1223 494 484; Email: juan@ebi.ac.uk.

**Short title:** Ten Simple Rules for taking advantage of GitHub in bioinformatics

**Introduction**

Bioinformatics is a broad discipline in which the common denominator is the need to produce and/or use software that can be applied to biological data in different contexts. For instance software is routinely needed for the analysis, visualization [PMID:25475079] or storage of biological information. For enabling and ensuring the reproducibility of scientific claims, it is essential that, together with the scientific publication, the corresponding dataset/s are made publicly available to the scientific community [PMID:24763340,PMID:25158685]. In addition, all the software used for the analysis should be either well described (e.g. in case of commercial software) or openly shared, when possible [PMID:24675742,PMID:25993922]. At present the latter is becoming more common in the case of “home made” programming scripts or for bioinformatics open-source projects, where the source code is made freely available and can be reused by third parties [PMID:25071829]. The availability of the source code can be achieved thanks to the existence of several code repository-hosting services such as Sourceforge (<http://sourceforge.net/>), Google Code (<https://code.google.com/>) and GitHub ([https://GitHub.com/](https://github.com/)), among others. These resources are also essential for collaborative software projects, since they enable the organization and sharing of programming tasks between different contributors to the same project.

Here we aim to introduce the main features of GitHub and related tools and services, as a web-based platform, which offers a free and integrated environment for hosting the source code, the documentation and the web page of open source projects. Different to other alternatives such as Google Code and Sourceforge, GitHub is not only a source code repository for programmers and software developers; it offers a complete suite a services and tools for version control of documents, source, data, and also a collaborative environment for research and international teams [add citation]. The cornerstone of GitHub is the well-known and open-source version control system Git, designed and developed by Linus Torvals in 2005 to control and extend the Linux kernel development; and has become the most widely adopted version control system; adopted by major companies like Google, Facebook, Twitter. GitHub is free to use for use for public projects and offers paid plans for private repositories, and it is the host of millions of open source projects. We will provide a set of recommendations for taking full advantage of the core features of GitHub to facilitate and manage the work on a given bioinformatics project and increase its profile to the scientific community. The manuscript is timely due to the announcement of the closure of another widely used service, Google Code. Therefore, many of the users of Google Code will therefore migrate to GitHub in the coming months.

**Rules**

**Rule 1. Structure your research projects: users, organization, repositories, and teams**

GitHub is structured around four major concepts: users, organisations, repositories and teams. Users are the keystone of GitHub; as any other social network such as Twitter, Facebook or Google+. Every user has a profile where they upload projects to share and connect with other users by “following” their accounts. Different to other source code repositories, it incorporates social functionality that makes a bioinformatican’s/developer’s identity and activities visible to other users. On the GitHub site, users create profiles that can be optionally populated with identifying information including: an image representing them, their name, email address, organization, location, and webpage. As a researcher you should make your profile visible to other users and displays all the repositories and organizations you are working on and a list of their latest activities on the site (Comment: here we can add a Figure 1 with major options of GitHub). Repositories (shorten term “repo”) are directories or storage space where your projects can live; it can be included inside you organization or can belong to particular users. Users can keep code files, text files, image files; data files, inside a repository; and while many users store programs and code projects, there’s nothing preventing you from keeping text documents or other file types in your project making GitHub and excellent service and platform for bioinformatics and open-source research [citation]. However, most of the users work in groups, teams, which provide a new concept: Organizations. Organizations simplify management of group-owned repositories (e.g. your research group code), are great for institutes, research labs, and large open-source projects that need multiple owners and admins. The main intention of Organizations is to group a set of repositories and developers under the dame “project” and they include: team-based access permissions and unlimited owners, administrators, and collaborators using teams (Figure 1). Finally, teams and collaborators define how every member of the organization will contribute with the project; it defines the permissions and the access level to the code and the resources within the organization/repository.

**Rule 2: Commits, Versions, Branches and Pull Requests**

The cornerstone of GitHub is the version control system Git, where everything from a small change in a text to a complete redesign of the software is controlled by versions [add citation]. Git thinks of its data more like a set of “snapshots” (versions) of a miniature filesystem. Every state of your research, data, documents, code needs to be updated to one of this “snapshots” using commits (Comment: perhaps here will be interested to add some snippet code of a commit). Every time the user commit (save the state of your project in Git), it basically takes a picture of what all your files look like at that moment and stores a reference to that “snapshot”. This is a main reason of the quote: “commit often, as most as you can; perfection later”; the member of the organization and repositories can at any time select the right version by going back to a previous one. However, most of the organizations/repositories has multiple teams, developers, bioinformaticians, and all of them making commits to the data make the process of controlling the changes difficult to manage. A Branch in Git is a lightweight movable pointer to one of these commits, where the team or the user decides he will achieve a milestone [Pro git citation, Chacon, Scott. Pro git. Apress, 2009. APA]. The most frequent branching pattern in GitHub is the master/develop approach where the master keep a pointer to the stable/release version of the repository and the develop branch keep a pointer to the development version where new features, data, information is added [citation].

In projects involving more than one contributor, everyone wants to be sure than the contributions of others increase the quality and move the project forward. Pull-Requests are an excellent tool for fostering versions review and if you're using Github for team projects, you should be sure of using these extensively. A good practice of integrating the code in the *master* branch is for someone else to merge your code into it, ensuring that at least two contributors review each feature, data, file or new change. Every small change in large/collaborative project should be done by pull-request boosting the quality of the project and its data.

**Rule 3. Let’s others contribute and add ideas to your projects.**

Issues are a great way to keep track of tasks, enhancements, and bugs for your projects but also a way to approach your audience and your potential contributors. Most of the previous repositories only provides and encourage the use of issues as bug tracker of projects. GitHub’s issues tracker has its own section in every repository, and can be use to trace bugs, new ideas, enhancements by using a powerful tagging system of each issue. Its issue tracking is exceptional because it is focus on collaboration, references, and excellent text formatting by using for each issue: (i) a title and description, (ii) color-coded labels help you categorize and filter your issues, (iii) milestone acts like a container for issues (e.g. weekly discussion 9/5-9/16 related with datasets), (iv) one assignee responsable for working on the issue, and (v) comments that allows anyone with access to the repository to provide feedback related with the issue.

A well-organized and tagged issue tracker will enable to upcoming contributors and users of the project to understand the project. The issue tracker tells the story of the repository/project where new users can learn the reasons behind decisions and contributors can discuss the progress of the project.

**Rule 4: Always ready to use: Tests, Deployments and Continuous integration**

The first rule of software development, especially in agile software development, is that the code needs to be ready as soon as possible [PMID: 25071829, <http://arxiv.org/abs/1311.7011>]; including main functionalities free of bugs, well-documented and tested. In 2005, Martin Fowler defined the basic principals for continue integration in software development (Comment: Fowler, Martin, and Matthew Foemmel. "Continuous integration." Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf (2006)). These principles have become “THE” set of Continuous Integration best practices, and provide the framework about how to deploy software and in some way data. Every repository, scripts, mathematical model, function should contains a set of self-automated tests; a source code my run, but that doesn't mean it does the right thing. The simple use of those self-automated tests is to detect possible bugs introduced by new features, or changes in the code or dependencies, but also detect wrong results or so called “logic errors” where your source code produce a different results compare with what you intended it to do [add citation]. Then, continues integration provides the way of automatically run all of these tests in the repository by checking dependencies, data, and environment dependencies.

To do Continuous Integration automatically GitHub has been integrated with other systems such as Travis (<https://travis-ci.org>), a hosted continued integration platform that is free to all open-source projects hosted on GitHub. These builds/deployments are triggered automatically when each developer checks/commits their code/data to the repository. Continues integration via Travis notifications, allows the final team and contributors to know if the new changes work and do not introduce any error in the repository, making the repo always ready to be use.

**Rule 5. Self-Hosted documentation, blogging and manuals using markdown and GitHub web-pages**

Providing well-written documentation helps users/collaborators understand, and contribute back to your project. A strong system used to serve documentation can make life easier for the team writing it and boost the productivity and visibility of the organization. The difficult step about documentation should be which specific test we will include, which examples are the most clear for the users; not configuring tools or figuring out how to deploy updates.

**Rule 6. Make your code easily citable by others in publications**

In research, it is always a good practise to be able to cite items using permanent and unambiguous identifiers. It routinely happens for proteins, genes, metabolites, different types of datasets, or published references, among many other items. Digital Object Identifiers (DOIs) have been used for many years as unique and unambiguous identifiers for enabling the citation of scientific publications. More recently, a trend has started to produce DOIs for other types of scientific outputs such as biological datasets [PMID:24727771] or training materials. The main motivation behind is to give scientists a better credit for their work [PMID:19587644], enabling at the same time a better way to cite and track it. Following this trend, GitHub now enables the use of DOIs to cite the code deposited, using the data archiving tool Zenodo (<https://zenodo.org/>). The procedure is very simple and it is explained in detail here ([https://guides.GitHub.com/activities/citable-code/](https://guides.github.com/activities/citable-code/)). By default, Zenodo takes an archive of your GitHub repository each time you create a new release. Before Zenodo can issue a DOI for your repository, you will need to provide some metadata information about the archived GitHub repository. Once the DOI has been assigned, apart from using it in your CV or add it to information resources such as Europe PubMed Central [PMID:25378340].

**Rule 6. Always link and highlight your deposited source code in publications**

As already mentioned in the introduction, reproducibility of the scientific claims should be enabled by providing openly the software and the datasets that are used in a particular study described in a publication. You should always highlight as much as possible in your publications that your code is freely available in GitHub, together with any other relevant piece of information that you may have deposited.

In our experience, this openness definitely increases your chances of getting the paper accepted for publication. On one hand, journal editors and reviewers have the opportunity to reproduce your findings during the manuscript review process and realize that you have nothing to hide. On the other, once the paper is published, the same can be done by any member of the scientific community, which can increase your opportunities for further discussion and collaboration.

**Rule 6. Build up your scientific career**

To move ahead in your scientific career or simply to find a new job, it is very convenient to show and demonstrate that you have contributed to a given project and at which level. In GitHub it is easy to show the level of contribution of individuals to a particular code base. It is then possible for third parties to appreciate whether one given person has been one of the main contributors or does only sporadic contributions, or to which projects someone is contributing in a regular basis. This can be very useful information for a prospective employer.

Another complementary way to build up your CV is to use GitHub to contribute to the increasingly used “altmetrics” methods to capture the impact of your work. In scientific publishing, altmetrics are defined as non-traditional metrics that have been proposed as an alternative the traditional citation impact metrics [23538811;25423184]. Altmetrics cover not just citation counts, but also other aspects of the impact of a work, such as how many data and knowledge bases refer to it, article views, downloads, or mentions in social and news media. One way to achieve this is to use the service “ImpactStory” (<http://www.impactstory.org/>). ImpactStory is an open source, non-profit organisation that provides web-based tool that provides altmetrics for blog posts, datasets, reviewer activity and software, among others. Users can link their ImpactStory profile to their GitHub account, displaying information such as the popularity of the project, or the number of *forks* (repository copies) that other people have made of your code. At present, ImpactStory is a subscription-based service. However, users can try the service for free for 30 days.

**Rule 8. Explore new ideas and discover the power of open source projects**

One of the jobs of a scientist is to read routinely the published literature available. Analogously, one of the jobs of programmers should be revise publicly available projects and code that can be interesting for his/her interests. GitHub makes this possible

Explore projects

Fork projects

**Rule 9. Be social.** Follow projects, people, projects, and issues.

You should try to learn as much as possible from your peers and colleagues and be updated with the developments of a project you are interested in. GitHub enables functionality that is already common is commonly used Social Media platforms such as Facebook or LinkedIn.

**Rule 10.** **Keep going!**

If you have never used GitHub, especially if you don’t have experience in using a versioning control system, there is the typical learning curve that can be applied to everything new someone has to do. However, in time, you will realize of all the time that you have saved by using GitHub.

**Conclusions**

To summarize, you can benefit for using GitHub in many different ways.

**Acknowledgements**

Y.P.R is supported by the BBSRC ‘PROCESS’ grant [reference BB/K01997X/1]. R.W. is supported by the BBSRC ‘Quantitative Proteomics’ grant [reference BB/I00095X/1]. J.A.V. is supported by the Wellcome Trust [grant number WT101477MA].

References

1. Wang R, Perez-Riverol Y, Hermjakob H, Vizcaino JA (2015) Open source libraries and frameworks for biological data visualisation: a guide for developers. Proteomics 15: 1356-1374.

2. Goodman A, Pepe A, Blocker AW, Borgman CL, Cranmer K, et al. (2014) Ten simple rules for the care and feeding of scientific data. PLoS computational biology 10: e1003542.

3. Perez-Riverol Y, Alpi E, Wang R, Hermjakob H, Vizcaino JA (2015) Making proteomics data accessible and reusable: current state of proteomics databases and repositories. Proteomics 15: 930-949.

4. Vihinen M (2015) No more hidden solutions in bioinformatics. Nature 521: 261.

5. Osborne JM, Bernabeu MO, Bruna M, Calderhead B, Cooper J, et al. (2014) Ten simple rules for effective computational research. PLoS computational biology 10: e1003506.

6. Leprevost Fda V, Barbosa VC, Francisco EL, Perez-Riverol Y, Carvalho PC (2014) On best practices in the development of bioinformatics software. Front Genet 5: 199.

**Figures.**

**Figure 1.**