14 Nov 2021

# Github workflow for conducting research projects

by Harpo Maxx

> Thanks to Juanma Romero, Tincho Marchetta and Rodralez from LABSIN for point me to most of the rules and procedures described in this post.

## The common scenario

Using **Source code Management** (SCM) tools for conducting a development process is basically the way to go in the software industry. Let be me clear: *Not using such tools is basically insane!* SCM tools are usually integrated into some kind of web application and they provide much more than just simple collaborative access to code and code versioning. You usually have access to some sort of ticket tracking system, a continuous integration (CI) tool, among others. Github, GitLab, and bitbucket are good examples of these kinds of tools. (I'm pretty sure you have heard about them 😜).

The Software Industry has well-defined standards and procedures which are heavily based on tools such as Gitlab. The Github flow, for instance, is a good example of some of the procedures applied in collaborative software development. However, in research sometimes we follow a more relaxed and not structured way to do our job. Don't get me wrong, of course, we use SCM tools (sometimes) for keeping track of our code. But, since research could be a solitary task, we tend to relax the standards. Every researcher is free to work the way she wants. Usually, the research topics are so complicated in their own way, that forcing the use of a procedure or specific tools is left aside. Such behavior could a big problem when a new member wants to join a particular project. An infamous research scenario is someone trying to join a project and not finding the last version of a particular model, nor the code that implemented it.

> ***The data capture process, the data analysis is certainly rigorous, the software... well that is a completely different story.***

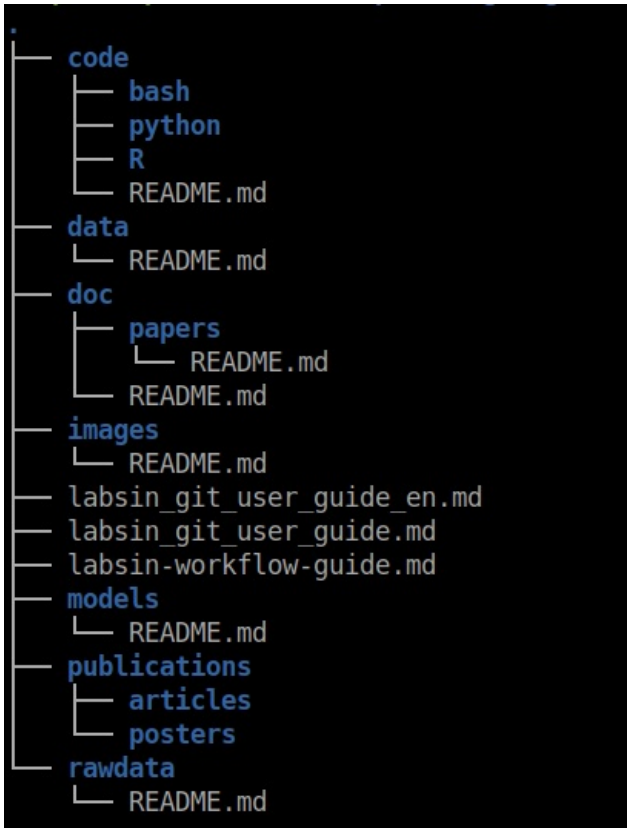
If you think a little bit about it, you can say it sounds ridiculous, a researcher that doesn't follow a rigorous methodology? Well, the data capture process, the data analysis is certainly rigorous, **the software**... well that is a completely different story. This behavior has started to change with journals and conferences asking for the code. More recently, sites like Papers with Code have started forcing researchers to be more careful when coding.

At LABSIN we have started exploring the application of these tools as a fundamental part of our daily research. In the rest of this post, I will discuss how we applied several ideas from software development to our research process. So if you have experience in the software industry this post could be a little be boring for you!

## The recommended repository structure.

Every new research project should have a repository on Github. Depending on the kind of research project, the Github repo could be **private** or **public**. The general idea of the repository is to include all the data related to the given research project. Not only the code but also all the resources used in the project. So you need to include a link to every important paper, book, video, internet article useful for the project in the repository. The contributions (aka publications) should be also included in the repository. Not only the PDF version but also the source (latex, word, markdown, etc.)

In the figure below, you can find the preliminary directory structure template we use every time we start a new research project. Please keep in mind this is very data-science oriented.




In the `rawdata/` directory, you should put the data used for the project without any kind of transformation. Since raw data used to be huge, you can simply create a file pointing to the datasets. In case data is stored in some kind of database, then the instruction for accessing the database should be available here.

If you decide to carry out some kind of cleaning, data transformation, etc.), or maybe you want to sample a smaller portion in the data for working in-memory, such data should be stored in the `data/` directory.

In the `code/` directory, you should put the source code for every experiment, study, and developed application related to the research. Since we basically focus on Data Science projects our main languages are `Python`, `R` and `Bash` . How to continue the hierarchy below the language directory? It depends. I personally like to create a new directory for `notebooks/`. Another for `scripts/` and (if necessary) one more for R `packages/` or`shiny/` apps.

The `doc/` directory is for all the references used in the project: Books, papers, datasheets, articles, videos, etc. For that, you can use a BibTex file for papers and books and a markdown file for the remaining resources. (In some cases you can include the PDF ).

From time to time you share a chart with your co-workers or a table with partial results. It is a good idea to save them. You never know when you will need it again. For storing these kinds of resources you can use the `images/` directory.

In the `publications/` directory you can store all the publications of the project. We use special directories for `posters/` and `articles/`. Remember, the idea is to include here all the sources of the publication for further modification.

Finally in the `models/` directory, you should store the results models. Here you can put the plain model object for use when new data become available. Very often models are deployed via some kind of microservice (Flask, fastAPI or Plumber for instance). in that case all the code of the microservice should be stored here (as well as Docker files if they are needed).

# The ticket system Slack Integration

# The Github flow

As I mentioned before, there is nothing new about this methodology

### We work hierarchically with 2 branches simultaneously:

`main`: Production branch. Nothing is merged to `main` except there is a strong consensus among all the participants working on the repository. Only the `develop` branch can be merged to the `main` branch.

`develop`: it is the integration branch used by all the branches to merge particular changes.

### For each new feature, you create a new topic branch.

These branches should always be created from `develop` and the idea is to have a local branch first where you work in the feature of the project (a portion of a paper, a new function, a modification to the Bibtex file, etc.) Then, you must create a Pull Request (**PR**) at the Github site, wait until other members approve it. Finally, when the PR is approved, the

merge between `develop` and your topic branch is made at the Github site. Finally, if all issues were resolved related to the branch, the branch can be deleted.

Categories

data-science machine-learning-engineering tools

Tags

reproducible-research