The Best Practices for Project Management

Suggestions for a re-producible and productive research project

Posted on January 7, 2017

As computational/statistical researchers, we often found ourselves had a hard in finding data, codes or related notes. From several years of research experience, I experimented many note-taking and project management tools. Among those, I found the following practices are the most efficient and productive ones.

Git controlled

Of course, it should be git controlled and better to be put on Github. Git enables us to work collaboratively and to track any changes from anybody. I would suggest all my coworkers put their research codes on github.

To store data, especially large data sets, I would suggest put them in shared data folders among lab members and put a symbolic link in your current project. Or creat a `largedata` folder in your current project and `.gitignore` the folder. Github offers a handy collection of .gitignore files, some of which are global and can be added to your global .gitignore, and others which are project specific.

Commit Messages

Use informative commit messages.
Read the following suggestions:

1. How to write a git commit message
2. On commit messages

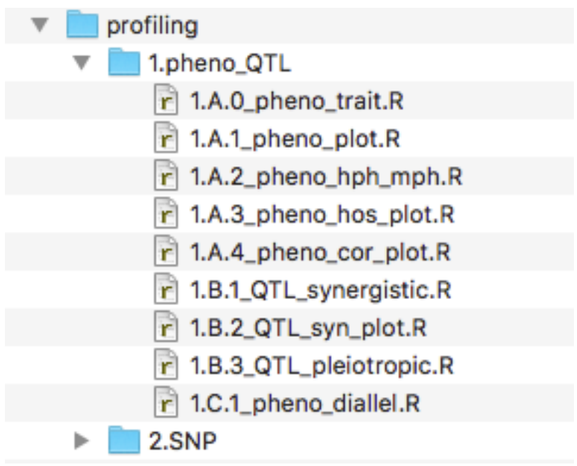
Project Directory Architecture

In a typical `R` project, I will copy the following folder into the project dir. The layout of directories is based on the idea from ProjectTemplate.

1. **cache**: Here we store intermediate data sets that are generated during preprocessing steps.
2. **data**: Here we store our raw data of small size. Data of large size, i.e. > 100M, store in a `largedata` folder that has been ignored using `.gitignore`.
3. **doc**: Documentation codes (i.e. Rmd files) for generating the figures.
4. **graphs**: Graphs produced during the analysis.
5. **lib**: Some functions used within this project.
6. **munge**: Here we store some preprocessing or data munging codes.
7. **profilling**: Main scripts for the project. It contains some sub-directories.
8. **TODO**: A todo list, markdown file.
9. **README**: readme file.

Numbering system for codes ordering

To manage research codes, I employ a numbering system. As shown in the below example from one of my research projects, I have multiple subfolders in the `profilling` folder. Codes were named by number, letter and other numbers separated with dots. My research was conducted according the order, i.e. I processed phenotypic traits using `1.A.0_pheno_trait.R` and then plotted the data using `1.A.1_pheno_plot.R`. It will always easier for me to go back to re-visit some of the codes or re-plot the figures with this numbering and codes ordering system.



Tags: Github, R, project

← PREVIOUS POST

NEXT POST →

Using Github for a Collabrative Research Project

fork, pull, push, and pull request

Posted on January 10, 2017

Getting started with Git and GitHub, check this out.

Here is the collaborative workflow:

1. `Fork` the project on Github to your own repo and `Git clone` or download source code.

```
git clone git@github.com:YOUR_GITHUB_ID/PROJECT.github.io.git
```

2. Create or edit files.

Folders in a project serve for their own purposes. Read this post to learn more about which folder you should put your codes in (i.e. `profilling/`) and which folder you should store intermediate data (i.e. `cache/`), and others.

Importantly, do not use absolute path. The path should be relative, i.e. use `profilling/1.code.R` but not `user/local/MYUSERNAME/Documents/profilling/1.code.R`. Because not all your collaborators share the same working directory with you.

3. Commit your changes

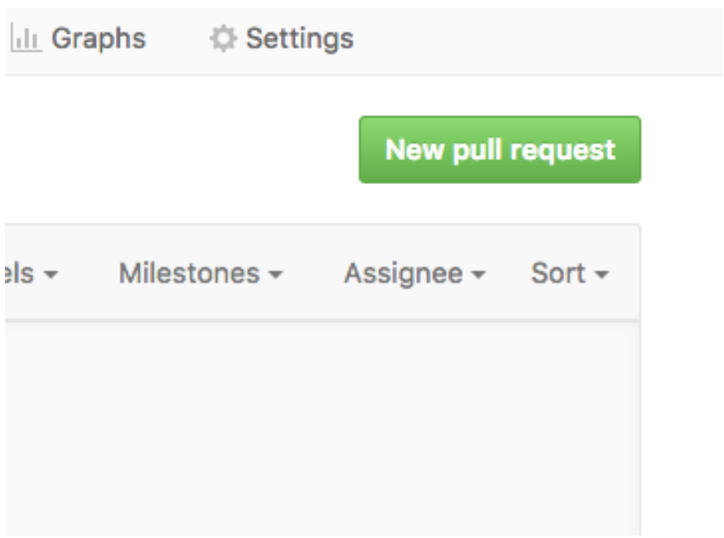
Commit with informative message. It is more informative to state why you do this, i.e. `to study the population structure` or `plot correlationship matrix`, rather than what you did, i.e. `added file1`. Because the latter can be figured out from the git histroy.

```
git add --all
git commit -m 'to explore pop structure.'
```

4. Push to the branch

```
git push origin master
```

5. Create a new `Pull Request`



Tags: Github, R, project

Computing Skills

Resources to learn basic computational skills

Posted on January 15, 2017

R Statistical Language

Online Courses

- HarvardX Biomedical Data Science Open Online Training

Free Online Resources for R

- The try R interactive online course from Code School
- The swirl: learn R interactively from within the R console
- Coursera: R programming class, taught by Roger Peng, Jeff Leek, and Brian Caffo
- Quick-R: Quick online reference for data input, basic statistics, and plots
- RStudio Cheat Sheets

R Books:

- Dynamic Documents with R and knitr By Yihui Xie
- R for Data Science by Hadley Wickham and Garrett Grolemund
- R for Everyone by Jared Lander
- R Cookbook (O'Reilly Media) by Paul Teetor
- R in Action by Robert Kabacoff

Git Best Practices

Python

Linux Command Lines

Tags: bioinfo, CS, tools

← PREVIOUS POST

NEXT POST →