Week 9 - Reproducible code and addressing code reviews

Based on the code reviews that you got from your student assistants, today is the time to improve your GitHub repositories and the code structure! The following list summarizes the most common errors that were made in the first weeks and how to correct them. Please go over them and you'll be soon finding yourself on the way to a clean and functional GitHub coding project with python! Try to include everything you learn today directly into your BIO-210 project on GitHub (for all releases starting with v3; if you already released it then release an update, e.g. v3.1)!

Make sure to release your code by 10am next Monday (release notes)!

1. **Addressing code reviews (issues)**

   The student assistants opened various issues during their review. When you commit or do a PR you can link to those in order to indicate what change addresses what issue. E.g. in a commit message you can write "git commit -m "closes #3", which will literally close issue #3 when you push it to GitHub (see e.g. https://github.com/DeepLabCut/DeepLabCut/issues/1150). You can also cross-link by putting hyperlinks. E.g. check out this example: https://github.com/DeepLabCut/DeepLabCut/pull/1582

2. **How to exclude file types from git repository**

   Add all file types to the **.gitignore** file that you don't want to track (e.g. notebook checkpoints, pycache, files from your IDE, virtual environments). See here for the documentation (https://git-scm.com/docs/gitignore) and check out also this example: https://github.com/EPFL-BIO-210/demo-project/blob/main/.gitignore. Note that if you add the **.gitignore** file after files that you want to ignore are tracked, then you need to "manually" remove them to also get rid of them (i.e. delete them, then stage and commit; you can also manually untrack with `git rm --cached <file>`).

3. **How to organize your python environments (with conda)**

   For reproducibility, keeping track of dependencies is important. An easy way is to use conda environments, you can create one isolated python environment for every project and install all needed dependencies into those. Go through the following guide (https://conda.io/projects/conda/en/latest/user-guide/getting-started.html) for conda environments, you can also look at the anaconda tutorial of week 3: https://github.com/EPFL-BIO-210/BIO-210-CourseMaterials/blob/main/docs/week_3.md. To make the environment you have created for your project reproducible, you will need to export an `environment.yml` file. A guide for that can be found here: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#sharing-an-environment.

4. **What to write in a good README file?**

   A README is a text file that introduces and explains your GitHub project. It contains information for understanding what the project is about and how to run code. Read this post (link) for what to include in a good README. Formatting READMEs is based on Markdown language, which enables you write nicely formatted and visually appealing texts! See here (link) for how-to write markup. Also check out some example projects: (https://github.com/DeepLabCut/DeepLabCut/blob/master/README.md, https://github.com/EPFL-BIO-210/demo-project and https://github.com/psf/black).

Search for different public GitHub repositories on your own, compare them and find out what makes a good and clean structure that helps for reproducible code and experiments!