Data Hygeine

For more please contact me at benucci[at]msu[dot]edu\n"
and visit https://github.com/Gian77

What is data Hygeine?

In general, data hygiene refers to **all of the ongoing processes involved in guaranteeing data is clean**.

Dirty data is data containing errors, whether it's outdated, incomplete, duplicated, or simply incorrect. These errors can be introduced at any point while that data is in your system, whether it was incorrectly entered initially or an accidental change was made when updating your records.

To reduce the amount of data errors in our shared lab space, we should adopt the following data Hygeine guidelines. Remember, omics data are for yourself, who generated them, but they are also for the whole scientific community when they are released to the public. The following data hygeine guidelines will help to make your data FAIR by improving Findability, Accessibility, Interoperability, and Reuse.

Useful tips to keep your data tidy

In the HPCC lab space

1. Create a personal directory in the "bonito_lab" space

- name your directiory using only your last name - no nicknames or abbreviaitons.
- change the access permision using `chmod 750 -R <directory name>` or `chmod 755 -R <directory name>` if you want to keep files just for yourself or to give read and execute permission to the group. This avoid accidental removal by other lab members.
  To learn how Linux permission system work, please look online or check out this link.

2. Keep a copy of the raw data in one of the `data_...` directory present in the "bonito_lab" space. Those are accessible by everyone on the group so be carefull do not erase them. Different data directory are present for different data types, for example:

```
drwxrwsrwx  9 benucci  bonito_lab 8.0K Nov  9 09:29 data_amplicons
drwxrwsrwx  3 benucci  bonito_lab 8.0K Jun 28  2022 data_genomes
drwxrwsrwx  3 benucci  bonito_lab 8.0K Jun 28  2022 data_metagenomes
drwxrwsrwx  3 benucci  bonito_lab 8.0K Jun 28  2022 data_transcriptomes
```

> **WARNING !**
> To avoid erasing files accidentally you can set up an alias to the rm command, in your .bashrc, so that prompts you before erasing. nano your .bashrc, then paste the alias at the bottom of the file as follows.

> `nano .bashrc`
> `alias rm="rm -i"`
> press "control+x", overwite the file, and then source it.
> `source .bashrc`

3. Please be organized and remove duplicated or intermediate files when you can. See below!
4. All databases will be kept in a directory called `DATABASES`. The directory will have x and r permissions so everyone can read its content. A person will be the maintaner of the DATABASES diretory so any additional database needed should be downloaded by that person.

How to set up your project dir

Create directories

An organized project directory structure contains several directories to store, code, data and results as tidy as possible. My sugegstions are:

- Create a directory for your project that contains the type of the data are you going to analyze, two/three words description of the project, and the date when project has been created, for instance:
  `mkdir metagenome_SwitchgrassRainShelters_011723`
  `mkdir amplicons_TransgenicSwithgrassMicrobiome_042022`
- Within the project directory you should create:
  - a main read me file, I like markdown format so it is github friendly `touch README.md`
  - a License if you intend to publish the code somehwhere, always good to have. I liek the MIT license. `touch LICENSE.txt`
  - create all the other directories for the project. The content should be included in each of these directories is selfexplanatory.
    `mkdir code`
    `mkdir documentation`
    `mkdir outputs`
    `mkdir slurms`
    `mkdir rawdata`
    `mkdir testing` or `mkdir debugging`

  > **NOTE**
  > To make sure that all these directories are good to go in github, do not forget to add a `README.md` in each of them. Remember, GitHub keeps track of files not directories.

- Another file you want to have, if you are planning to push your project directory on GitHub, is the `.gitignore` file. In this file you should include all the files you do not want to be track by git. For example, .fastq files.

General rules on files and workflows

The code

A good idea is to number your script so they follow a logic, for example, from 001 or using sections a001, b001 and so one. Depends of your project and what are you trying to do.

The output

Once you have your project directory created and set, then if is up to you on how to manage it. Some people (like me) like to add subdirectories, for example within the output directory, for each of the obtained results/tool tun.

At the end of an analysis, you should keep just the start (the raw data) and end (the results data) data. ALl the intermediate files, once sure that aren't needed, should be erased.

All the files that are kept should be compressed to save storage space. A good compression method is `tar.gz`. For example, if I was to compress an entire directory and all files within it, and remove the original files, I can do:
`tar -zcvf mydir.tar.gz --remove-files mydir/`
See the manuals `man tar` and `man gz`.