REPEATABILITY/REPRODUCIBILITY

Repeatability = ability to reproduce the same results using the same reagents over time. This is quite difficult in biology as well as in other fields of research.

High-throughput data are stored in a FASTQ format in online repositories.

Results of the analysis of differentially expressed genes won't be the same when repeated even if we start from the same raw data.

Standardization in this context is paramount to enhance repeatability, but tools continuously improve for instance and thus it may be not possible.

Ironically, you may use two different biological approaches and obtain the same results, and on the other hand use the same protocols and obtain results that can't be superimposed.

Another issue linked to repeatability is that we don't always find in literature all the information needed to reproduce an experiment because authors omit to give details about it. **It is important to report everything that is necessary to reproduce the experiment you carried out**, so that other people can repeat the same protocol and find mistakes or highlight different outputs in general.

Scripts allow to track down possible mistakes in the data analysis and avoid publishing wrong results (differently from what you can do with Excel, for example). **A script enables to reconstruct the whole analytical procedure**.

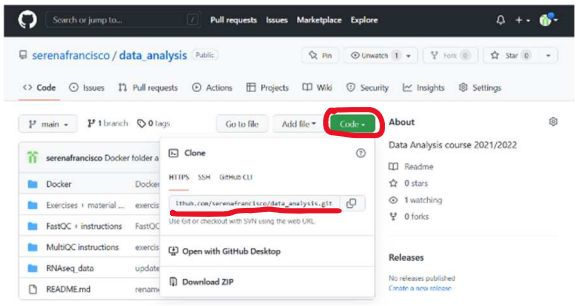
Rules to improve reproducibility in computational research:

1. Keep track of how every single result was produced
2. Avoid manual data manipulation
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results (in standardized formats)
6. For those analyses including randomness, note underlying random seeds
7. Store raw data that are behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

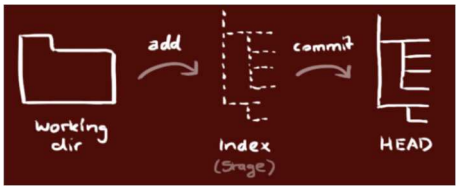
GITHUB

GitHub is a code hosting platform for version control and collaboration. It allows you and other people to work together on projects from anywhere. Every repository created on GitHub should include a **README.md** file, which is a Markdown file summarizing the work you're doing. At the exam we'll be required to write a README report in a clear .md format → Markdown is a syntax that makes ASCII characters look better with some basic commands (https://www.markdownguide.org/basic-syntax/).

After you've created a GitHub account and a (remote) repository, install the git software (https://git-scm.com/downloads) and then create a local clone of the repository on the GitHub browser page (remote repository). Type on your terminal `git init` to initialize your local GitHub repository. Now you can clone the remote repository: on the GitHub website (of your repository), click on the given button and copy the link:

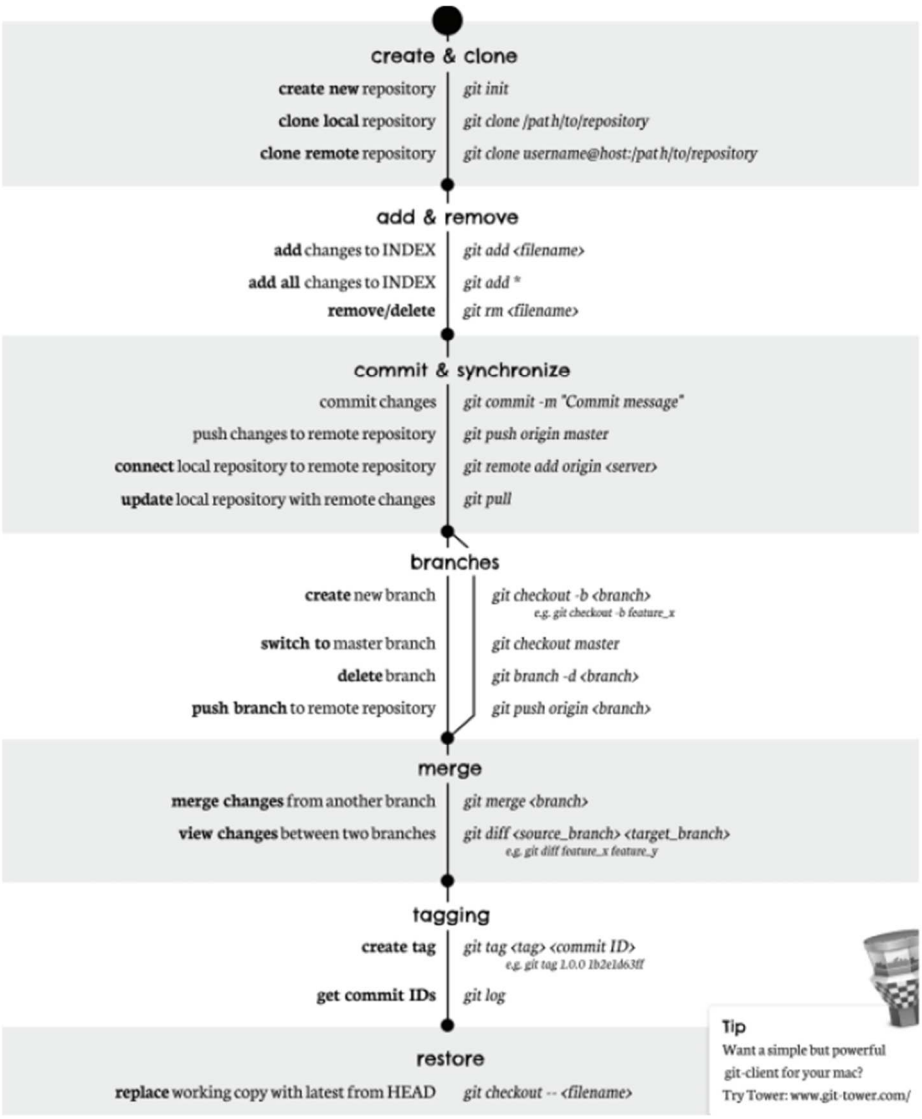


On the terminal type: `git clone https://github.com/serenafrancisco/data_analysis.git` → you just cloned your repository (remember to do it in a specific folder on your computer).



From now on, if you modify the repository from your computer (locally), you'll then have to commit the changes also on the remote one. To do so, 3 steps are needed because the local repository consist of 3 "trees" maintained by git: the working directory (which holds the actual files), the index (staging area) and the head (which points to the last commit you made).

1. `git add *` → to add all the modifications done, on multiple files, from the working directory to the index. If only specific modifications have to be added to the index, then replace * with the name of the file(s). Basically you save the changes to the index structure to prepare files for upload.
2. `git commit –m <text of the message>` → to apply to the remote server the modifications you just added and make them permanent (i.e., modifications sent to the head, this way only what you've modified will be updated). To modify in a permanent manner the repository needs a reason, so a message is required to explain why and what you did (specially when working with someone else).
3. `git push origin <name of the repository/branch you want to update>` → to push the head to the remote repository or a chosen branch of it. You just updated the remote repository (finalization).



BASH COMMANDS

https://ubuntu.com/tutorials/command-line-for-beginners#4-creating-folders-and-files

R STUDIO

Here you can write R scripts and execute them. It includes a **console** from where you execute commands and a window where you can write your scripts. Every time a new variable is generated, this will appear in the **environment**. From the bottom right panel you can see the folder you want (click on the 3 dots on the right), sort of file explorer integrated in RStudio. By moving here, you don't change the directory you're located in: to do so, type the following command in the R console: `getwd()` to know where you are, and then `setwd()` to change the directory. If you click on "**Run**" (window of the script) you can see executed on the console the lines of command you just wrote. RStudio is easy for everyone sine it shows error messages and allows the correction of you script. When you want to save a script, this will have the .R extension and will be saved in the folder you moved to with `setwd()`.

How to update GitHub from RStudio → Modifications can be uploaded on GitHub from RStudio in the following way: *Project* (upper right button) → *Open a project* → open the file you want to be updated on GitHub. At this point you'll notice that there is a **Git** button (again, upper right panel) from where you can commit and push (↑**Push** button) to the GitHub remote repository.

Every time you don't understand an error message, try typing it on Google and see what's the solution.

DOCKER

Docker includes **containers**, which literally have the very same structure (they are piled up) and can contain any kind of product (also different stuff). This concept is applied to software, for instance those used for RNA-seq and scRNA-seq, that are quite sophisticated to be installed locally and also they generally work on Linux → **Docker allows to store a Linux software within a box (the container) that will work on a Windows or IOS environment**. For Windows there is the Linux emulator WSL, which reads what a container contains and executes it. So you don't have to install anything on your computer but you'll just need to run the specific Docker container. Moreover, the specific Docker container will always be the same, even if the hardware is changed, so every time you'll use it you will obtain exactly the same results if you start from the same data. **This warrants a certain level of reproducibility**. Docker is also better than a virtual machine as it doesn't require space on the RAM and disk. Docker software normally runs in background and uses RAM or disk only when it is actively computing. Space on the RAM is occupied only when a container is running. A Docker container includes the software, while the scripts used to run those software are in a GitHub repository with their description to allow to use them (and then data used for the analysis are stored in some online databases).

These are some docker commands to use on the terminal:

- `docker ps` → it tells whether there is any docker running at the present time. When a docker image is running, it is called docker **container** and has a specific ID.
- `docker run` → to run a docker locally. Anytime a docker is being run, if it's not located on the PC, it will be downloaded from internet: basically we give an image to Docker and if this image isn't found on the PC, it gets downloaded from an online repository (see the figure below).

```
PS C:\Users\seren\Desktop\dataset1> docker run -t ewels/multiqc
Unable to find image 'ewels/multiqc:latest' locally
latest: Pulling from ewels/multiqc
e6b0cf9c0882: Pull complete
536836f33c44: Pull complete
9fc26402a069: Pull complete
f8a5d905d402: Pull complete
4f4fb700ef54: Pull complete
1725e9e94554: Pull complete
Digest: sha256:cb9905df9c8c9b374aab05a687c2da729c40de318c3cf9f276a8076c0ad0a5ec
Status: Downloaded newer image for ewels/multiqc:latest
```

The image ewels/multiqc is not present on the computer. Thus it is downloaded online from the Docker Hub.

- **`docker run -t`** → whatever is happening will be shown on the screen.
- **`docker run -v`** → a connection between our hardware (PC) and Docker, by mounting a folder on our PC (where data of interest are located) to the docker folder. This way Docker can see what there is on our PC.
- **`docker run -w`** → indicates the directory where the docker has to be executed (where we'll find the results).
- **`docker images`** → it shows docker images located on the PC.
- **`docker ps -a`** → to show the docker that have been run and the ID of the specific container.
- **`docker logs`** → when the docker is running in background, to see the logs coming from the specified container (it gives info about what happened during the execution of a container of which we know the ID).

ewels/multiqc is a repository (image) on Docker Hub (online) → images come from Docker Hub and are uploaded by users (such as the user ewels in this case).

A **container** is a sort of virtual machine that allows you to run a software which wouldn't supported by the actual environment of your computer (e.g., Windows OS). The **image** per se is simply the software, while the execution of the software occurs inside the container.

An image is made of the software, which is based on its own script (default script). You can be good with this default script but if you want to have the software based on a different source of code, then you can write your own script which will be a Dockerfile: in this case type docker build + (see dedicated lesson).