1 - Introduction

TOPICS	Bash commands, CommitAddPush, Docker systems, GitHub, MarkDown, RStudio, Repository, repeatability, reproducibility
DATE	@March 9, 2022 2:00 PM
LAST EDITED BY	
LAST EDITED TIME	@May 1, 2022 4:27 PM
MADE BY	
PROF	Calogero
Recording	20220309 Data analysis Calogero 1.aac lesson1.mp4
STATUS	✓

This course will focus on R.

You have to install Telegram and join the group chat.

During the exam it will be possible to use all the on-line helps (copying from google is not a bad thing per se, especially if you are learning)

🚩 **The presence at lessons and the exercises are mandatory** every week.

Reproducibility VS Repeatability

During wet lab practice, many repetitions of the same experiment are needed to ensure that your results are reproducible. But actually, this is more properly a matter of **repeatability**, i.e. the ability that the same operator with the same reagents is able to obtain the same results over time. same location,same instrumentation

Reproducibility is due to the fact that **different people in different places with different reagents are able to achieve the same results.** This is really different in biology since it has a certain amount of intrinsic stochasticity which sometimes renders it impossible to get the exact same result.

Let's consider the example in which you do a KO of a gene and you obtain a lethal phenotype. If you repeat the KO and the phenotype is different, the experiment is not reproducible.

Reproducibility is an issue not only in biology: it is an issue in psychology, for example, and in all the topics concerning AI. A few years ago there has been a *reproducibility crisis (2007)* where some companies tried to reproduce experiments published by academics and they managed to reproduce only 60% of them.

Sometimes the paper does not give you all the information you need, but if you want to find subpopulations of differently expressing genes unless they give you all the information is practically impossible

Today high throughput data produced have to be deposited in repositories in FAST Q data format. If we consider the example of a fast Q file containing the output of an analysis on differentially expressed genes, for example, there are only a few chances that somebody else having the same raw data will obtain the same set of differentially expressed genes. Moreover, as the instruments get updated and improved, the statistics that are behind them change slightly and thus the final results will change. It is important to standardize also the instruments we use to make the computational analysis to gain reproducibility, but this may be a problem since tools evolve constantly. Unless using the exact same version of the tools you won't get the same results.

If we talk about gene expression, at the beginning of the microarray world (before RNA sequencing) the critical point was the fact that 2 independent experiments provided two completely different results. Depending on the setting you may get different results and only a very small part of them will be overlapping. The reproducible results are the most important part of our analysis.

If you do the same experiments with a different biological approach, you might get results that are biologically superimposed, even starting from different raw data. On the other hand, sometimes even using the same protocol and tools as the literature article did, the results are different.

Today reproducing results from one paper is important, but sometimes papers are not reporting all the information you need to reproduce the same analysis. For example, often the data presented in the article are not complete: if the data published are only above a specific p-value, it is possible that the rest of the gene list is not given; this means that you have to repeat the experiment to analyze the complete dataset.

Differential expression analysis is a simple analysis to do also on single cells. However, if papers do not report everything that they used to obtain the results, it is practically impossible to reproduce the experiments.

🚩 Critical point: report everything necessary to reproduce the experiment. Previous data is super useful to better focus the target goal for future analysis.

The aim of this course is to give the student a set of basic tools needed to guaranteed a certain level of reproducibility.

Nature Biotech Paper

In this Nature Biotech paper, researchers analyzed a number of cell lines treated with drugs to calculate the correspondent IC50. They identified resistant and not resistant cell lines and by doing differential expression analysis they obtained the list of genes associated with drug resistance. This approach was later demonstrated not to be reliable: since the data was available, further analysis on that paper could be done. The list they got in the first time was different to the one obtained in the repetition and also the heatmap of differential expression was not reproduced. This paper in the end got retracted.

Why did this happen?

Statistics analyzed this problem and tried to understand the cause. The problem was that the authors manually manipulated the data with excel. Something during the reorganization of the columns went wrong producing a misalignment between the data and the identifier, i.e. the name of the genes.

Manual manipulation of data is not something to do to retain reproducibility. Even basic procedures such as the renaming of files should be done with a **script to keep track of everything** and eventually go back if something is wrong. The advantage of writing scripts is that errors can be reproduced and caught because you have your script. You can correct it later. Even by renaming files you can also do something wrong which without any script would go undetected.

Hence, by writing the script it is possible to track down any possible mistake, something that is not possible using Excel. **The script** in brief **helps to reconstruct the path of the analysis.**

Ten Simple Rules for Reproducible Computational Research

From Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) - Ten Simple Rules for Reproducible Computational Research.

PLoS Comput Biol 9(10): e1003285. https://doi.org/10.1371/journal.pcbi.1003285

1. For Every Result, Keep Track of How It Was Produced
2. Avoid Manual Data Manipulation Steps
3. Archive the Exact Versions of All External Programs Used
4. Version Control All Custom Scripts
5. Record All Intermediate Results, When Possible in Standardized Formats
6. For Analyses That Include Randomness, Note Underlying Random Seeds
7. Always Store Raw Data 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

🚩 **Homework**

Part 1

Create a GitHub "data_analysis"

Write the description.

Create a README.md containing the name of the GitHub

Using line command create the local data_analysis repository (clone) mkdir RNAseq_data

Update the repository adding the folder RNAseq_data containing the file present in the course moodle Lezione 1: dataset esempio RNAseq File

Unzip the zip file downloadable from Lezione 1: example of scRNAseq File

Update the repository adding the folder scRNAseq_data with the fastq.gz files present in the Lezione 1: example of scRNAseq File unzipped file

Part 2

Install docker-desktop (https://www.docker.com/products/docker-desktop)

Install R (https://cran.r-project.org/)

Install RStudio

Install devtools library

Install docker4seq

Install rCASC

Check and learn the basic commands on GitHub, which will be useful to use later in R-studio

When you have a complex project you can build a GitHub for your project, this way you can split the project into multiple branches and then work on them in parallel.

This workplace also enables the recovery of previous versions of the same project, this way the possible errors can be found and removed.

Many people can work on the same data and then the 2 Versions are merged together with gitHub.

First steps in GitHub

On GitHub there are both commercial and not commercial projects, of course if you want to build something for sale, then you want to keep your work private. With the free profile only one private repository can be created.

1. **Create a GitHub account**
2. **Create your own GitHub repository**
    i. Create a new repository → new → public or private → choose public
    ii. Tick the "Add a README File" option.
    The README file is important for you to summarize your work easily for future reference i.e. why you wrote that script and what did it do etc. It can also contain images. It is a kind of diary,
    You save your script of the GitHub and you safe your results, in the readme you describe what you have done.

    Readme file is written in the .md (MarkDown) format.

> 🚩 **Markdown**
>
> Markdown is a specific and easy syntax to be used in .md files. There are some basic commands to make readme file more readable → Put in a pretty way the ASCII characters.
>
> Here you can find a link with all the basic syntax of a markdown document.
> https://www.markdownguide.org/basic-syntax/
>
> **At the exam, you will be required to write your README report in a clear markdown format.**
> **Fun fact: Notion is a MarkDown format text editor**

3. Then you want to create a local clone of the remote repository (the one on the github browser page): before doing this you need to
    - Install the git software there are different types of download depending on the OS.

- Do the initialization of the local GitHub repository (actually sometimes this is not really done because you go online, you create it and then you just duplicate it). This requires the command `git init` in the terminal.

Then you can clone:

i. Go to the main page of your new GitHub, there is a green button called "Clone" with a link to the position of the repository in GitHub. You need to copy it and give the command to the terminal.

ii. On the terminal, go to the working directory you want and then give the command `git clone <link to the repository>` . You have now cloned your repository. You will find a your .md README file in the directory where you decided to clone the git.

3. If you do modifications locally on your PC, then you need to upload the modifications also on the remote server of GitHub.

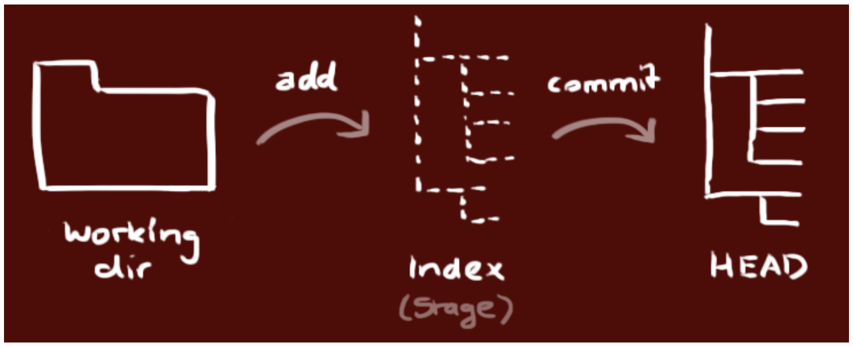
To do this you need to do three steps, because your local repository consists of three "trees" maintained by git (the working Directory which holds the actual files, the Index which acts as a staging area and finally the head which points to the last commit you've made.

The three steps are:

i. `git add *` → **add all and only** the modifications you have done on multiple files from your working directory **to the index**. If you specifically want to add the modifications you've done in a specific file you need to substitute the * (which stands for every possible letter, and hence allow you to add all the changes you've done on any file in your repository) with the name of the file.
This command is very useful since it enables to upload the remote repository just with the modifications that have been added to the file, thus it avoids the useless upload of extra data that would be repetitive.

committing impegnarsi

ii. `git commit -m "text of the message"` → **Commit** the changes: when you commit, you apply on the remote servers the modifications you've done. With the command commit, **you make the changes permanent**: commit updates the working directory in a way that can be uploaded in the remote server. To do this you usually need a motivation, and that's why it's good practice to explain briefly and keep track of what you've done with a message.



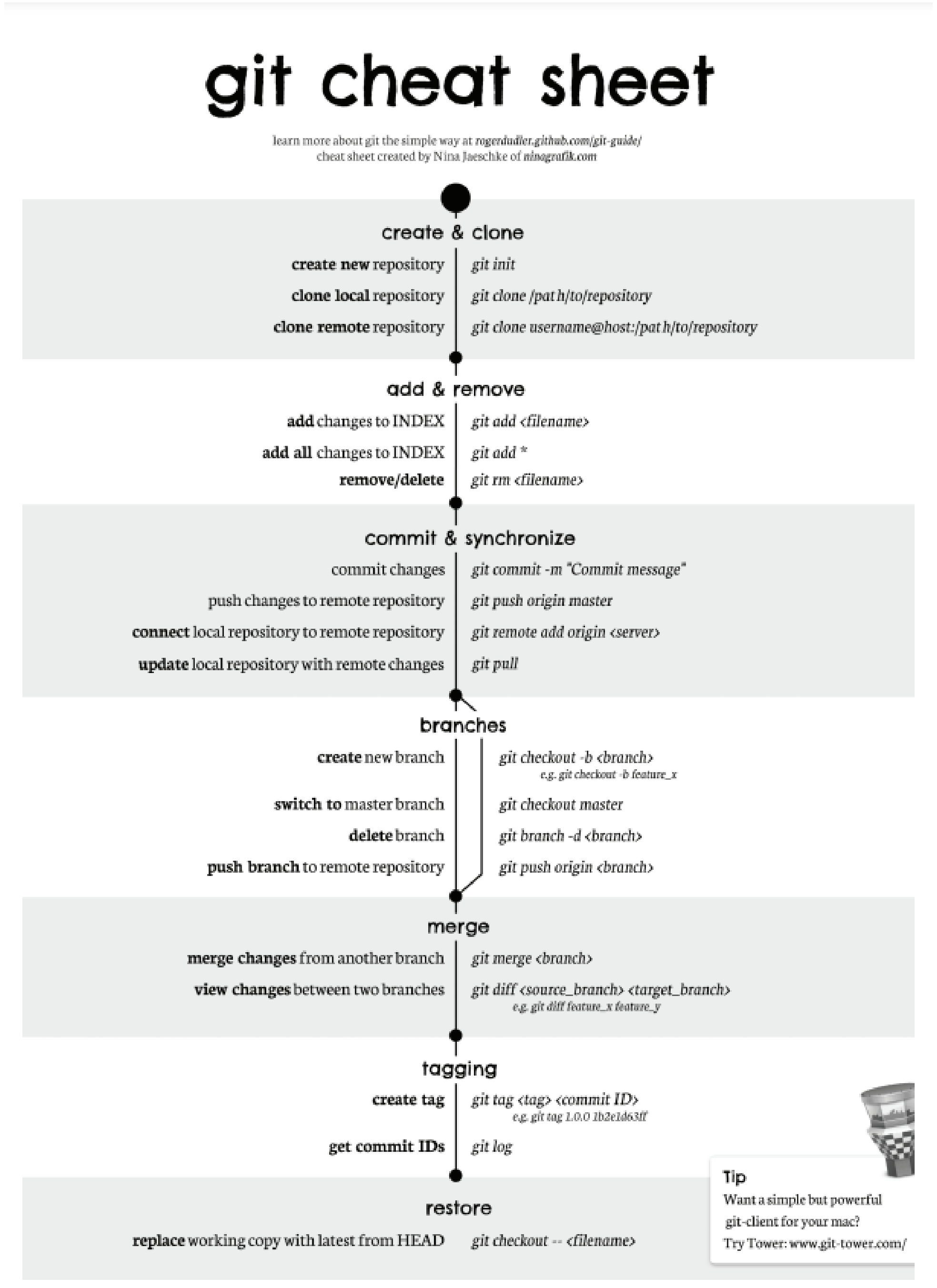
iii. `git push origin <name of the repository or of the branch you want to update>` → push the head to the remote repository or to the chosen branch of your repository. You have now updated your remote repository.

git push origin main this indicate you are updating to the main brach if you create new branches you can indicate them

⚠️ Path for the modification of your repository:

1. First you remotely modify the files.
2. Then with add you save the changes on the index structure, to prepare the files for uploading.
3. Then with commit you send it to the head (the main structure) and in this way you only update the things that you have modified.
4. With push you finalize the changes on your remote repository.

Other basic commands needed to work in GitHub



git cheat sheet

learn more about git the simple way at *rogerdudler.github.com/git-guide/*
cheat sheet created by Nina Jaeschke of *ninagrafik.com*

create & clone

create new repository	*git init*
clone local repository	*git clone /path/to/repository*
clone remote repository	*git clone username@host:/path/to/repository*

add & remove

add changes to INDEX	*git add <filename>*
add all changes to INDEX	*git add **
remove/delete	*git rm <filename>*

commit & synchronize

commit changes	*git commit -m "Commit message"*
push changes to remote repository	*git push origin master*
connect local repository to remote repository	*git remote add origin <server>*
update local repository with remote changes	*git pull*

branches

create new branch	*git checkout -b <branch>* e.g. git checkout -b feature_x
switch to master branch	*git checkout master*
delete branch	*git branch -d <branch>*
push branch to remote repository	*git push origin <branch>*

merge

merge changes from another branch	*git merge <branch>*
view changes between two branches	*git diff <source_branch> <target_branch>* e.g. git diff feature_x feature_y

tagging

create tag	*git tag <tag> <commit ID>* e.g. git tag 1.0.0 1b2e1d63ff
get commit IDs	*git log*

restore

replace working copy with latest from HEAD	*git checkout -- <filename>*

Tip
Want a simple but powerful git-client for your mac?
Try Tower: www.git-tower.com/

Image from the website he recommended.

Basic Bash commands

Here is a link with a good explanation of the basic Bash commands in Linux he showed last year. Be careful that with the windows terminal some of them are not working (ex: `ls` is replaced by `dir`). If it doesn't work on windows just google the correspondent command. https://ubuntu.com/tutorials/command-line-for-beginners#4-creating-folders-and-files “directory(sostituisci col nome) % ls” per vedere cosa c’è lì

- `ls` = list → show files located in the directory
- `ls -lh` → long human-readable extended representation of the files, with the size of the files, the permission that the files have.

  What are the permissions?

  You can look at the permission of a specific file at the beginning of its name when it’s listed in the terminal; the permissions are organised considering who can have access to the files: user-group-all.

  The permission appears as rwx = read, write and execute. Each group will have determined permissions.

  d = folder / directory

- `cd` = change directory (cd then space and then name of the folder. without any folder name it brings you to the main folder)

“cd Desktop” to move to desktop

- `cd ..` = brings you one level up
- `pwd` = tell me where I am
- `mkdir prova` = makes a new directory called “prova”

R studio

Go to Rstudio and download the compatible version for your PC.

Github is good to work in R-studio. R-studio is an infrastructure where you can write your R code, in the R Console, and it gets executed. It is also a sort of electronic lab notebook to keep track of your computational analysis.

As you open, you have an R **console** on the **bottom left part** where you see what you are doing (you execute). In the upper part you have the part where you can write your code.

⚠️ First write the code and then run and see in the console if you did something wrong. If this is the case an error message will show up.

Every time you generate a new variable you can see it on the right upper part, in the **environment**. For example, if in the R Console you write “x=1”, it will appear in the environment on the right.

Moreover, the lessons will show how there are a lot of useful information stored in this upper right panel.

The lower right panel gives information about where you are in the computer folders; by using the 3 dots on the right you can move to see the folder you want, for example the “Desktop/Hello World”. It is basically like a sort of “File Explorer” in Rstudio. By moving in this panel, you won’t actually change the directory you are in; to do so you will need to put a command in the RConsole, like the following:

- getwd() = get the working directory → to know where you are
- setwd () = to go in a new directory

✅ So, to summarize, on the right of the Rstudio space:

- “Environment” → information of the variable you added to the code.
- “Files” → tells you where you are

By clicking on the green “+” symbol on the upper left, it possible to open, create and modify script files by selecting the “R Script” option.
Basically, you will write the code lines in this file and then you “Run” them with the button →Run. Then, you will see the results of the running in the RConsole panel, which now is on the bottom left quadrant.

By clicking on →Run, the lines of code written are executed on the console.

⚠️ DO NOT write the command in the Console, but solely in the RScript

Moreover, RStudio is very helpful for not expert users, since it shows a pop up window while you are writing the command; this window contains suggestions for commands as well as error messages in case the command is not written correctly.

If you mistakenly run a wrong command, a red message of error will appear in the console and, if the error is not very complex, it will also explain the kind of mistake you did.

You can also save the file by clicking on the floppy disk icon near “Source on Save” . The file saved will have the extension .R and can be saved into the “Hello World”.

Then, since the file has been modified, it is necessary to upload the modifications on the GitHub; to do so there are two options: you can do it from the terminal like explained previously, or you can utilise directly RStudio.

To update to the GitHub server directly from RStudio, you can Open a project on the upper right button Project (you have to say “Don’t Save” in the pop up message that opens), go on the desktop and open “Hello world” as a project (the name “Hello world” will appear in the right angle of the Environment panel as a project).

Then from the “Git” button you can commit and push to the GitHub repository as before, but in an easier way. Thus, there are Git options on the upper right panel to help you with the commitment of the files on GitHub.

⚠️ Remember to write a message of commitment before committing the modifications

Once the commit is done, you can click on the ⬆️Push button and now the remote repository has been updated with the new modifications.

At the beginning you copy pieces of codes to do something you want, then you modify it and lastly you learn to write code. Google is the easiest way to retrieve code. You will be allowed also at the exam to google everything you want. All the online options will be allowed at the exam.

Errors are pretty normal, do not get scared by them, especially at the beginning: errors tell us what is wrong, so we can easily solve our issues and learn from them.

If you do not understand the error, cut and paste it on Google and you will find a solution to it. Normally the error are pretty trivial, like forgetting a comma or closing a bracket.

Docker engines/containers

Another problem that is generated from the problem of reproducibility is that if a certain number of software and libraries are installed on the PC, the user cannot be always sure that, by installing one libraries, the others will be updated. This is a critical issue.

Since few years a specific infrastructure, called **Docker**, has been developed to help with the organisation of the data. The **docker containers** represent exactly the same approach that is used to distribute goods using tucks, ships or airplanes with the physical metal boxes.

The physical containers have been built with a common structure that allows them to be piled up and also to contain any kind of product; the advantage of this system is that the same container can transport different things.

The same approach has been applied to the use of software; today, especially for RNA-seq and sc-RNAseq analysis, the problem is the software are quite sophisticated to install them locally and they work on Linux.

✅ Docker helps to store a Linux software within a box that will work on a Window/IOS computer and are constant through time.

The emulator of the Linux interface (WSL for Windows) will read what is present on these containers and execute. Thus, you don't need to install on the PC everything necessary to do an analysis, you will put everything inside a docker and utile is to run par of the analysis itself.

The docker box remains the same through time, so even if the infrastructure (hardware) is changed, the data obtained will be the same. For example, things stored in the docker now, will remain the same in three years and can be reused even if the PC is different.

Moreover, if you change your pc you can download the same docker and you have your software back and running even on different environments.

Dockers are one of the way that enable us to guarantee a certain level of reproducibility.

Normally, with a virtual machine, some part of the RAM and disk will be dedicated to it, e.g. if you install a Linux virtual machine on Windows.
When the virtual machine is started, those amounts of RAM and disk will be used by it even if the machine isn't actively working.

On the other hand, the Docker software normally runs in background and can execute its image as a normal window software, thus using RAM, disk...only when it's computing.
So, dockers are similar to a virtual machines, with the difference that they do not require a fixed RAM space on the computer.

Then, when the computation is finished, all the RAM and the space is free again. Thus, the space is used only if the container is running.

The advantage is that you don’t limit the space on your PC and hardware when you do not use the specific object (the docker in this case).

The dockers will occupy some space on the computer, but this is ROM and is relatively small compared to the requirements of virtual machines, which are very large.

> ❗ To run docker on Windows you need a very simple virtual machine that changes its dimensions depending on the amount of RAM requested by the docker to be used.

Normally, the docker engine is designed for Linux, thus if installed on a Linux or on a Mac (which is very Linux oriented), the helping virtual machine is not needed.

> ✅ Dockers are the way to guarantee reproducibility of the analysis

Example of docker application:

The data of one RNAseq experiment, stored as FastQ files on the GEO database, are analysed with Star2.5, RSem3.1 and DeSeq2.5.
Instead of installing on the PC all these software that will require much space, a docker container is built will all of these inside and put aside.
The docker can be also stored in public repositories.

The docker stay constant in time, thus it will give the same results if the input data is the same.

This is why dockers are very useful and guarantee reproducibility. They cannot be built at the beginning of the experiments, they are made after the pipeline is defined.

After building the docker, the same analysis is repeated to check if the results obtained are the same; then at that point, you can store the workload.

Dockers are also useful with reviewers that want to check out the software used in the analysis.

At the end, dockers of another published work can be used to analyse your own data. The container becomes a piece of software that can be used to run part of the analysis described on GitHub.

> 📌 The docker container has the software, the scripts used to run those software are in the GitHub with the description to use them. And then the data is stored in some other online database.