Introduction to Bioinformatics

Git for Bioinformatics Analysis, Project Management and Collaboration

Introduction

- When performing bioinformatic analysis and starting a bioinformatics project you’re expected to write codes
- Not only that, sometimes you also have to track the results of your experimentation for the analysis
- Let’s start with a story

First Scenario

Day 1

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

targets <- read.metharray.sheet(base_dir, pattern="SampleSheet.csv")

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

Imagine you're writing codes in R for your analysis, the code works perfectly and returned the expected results

Day 2

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

The next day you realize that you didn’t need some lines of code, so you remove it

Day 14

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

targets <- read.metharray.sheet(base_dir, pattern="SampleSheet.csv")

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

Fast forward to day 14 of you're writing code it turned out that you have to go back to your initial code

Day 14

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

targets <- read.metharray.sheet(base_dir, pattern="SampleSheet.csv")

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

Well for a shorter code like something above its easier, we know that we removed ann450k and target variable

But the truth is

```r
# Get all the CpG sites used in the analysis to form the background
all <- DMPs$Name
# Total number of CpG sites tested
length(all)

par(mfrow=c(1,1))
gst <- gometh(sig.cpg=sigCpGs, all.cpg=all, plot.bias=TRUE)

# Top 10 GO categories
topGSA(gst, number=10)
gst[,gst$ONTOLOGY == "BP"]
```

Turns out when you're doing bioinformatics project things can really get complicated

But the truth is

```r
# Get all the CpG sites used in the analysis to form the background
all <- DMPs$Name
# Total number of CpG sites tested
length(all)

par(mfrow=c(1,1))
gst <- gometh(sig.cpg=sigCpGs, all.cpg=all, plot.bias=TRUE)

# Top 10 GO categories
topGSA(gst, number=10)
gst[,gst$ONTOLOGY == "BP"]
```

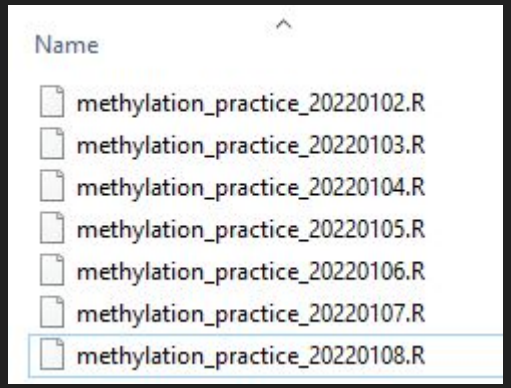
Number of files will increase significantly, so are the results and the processed data

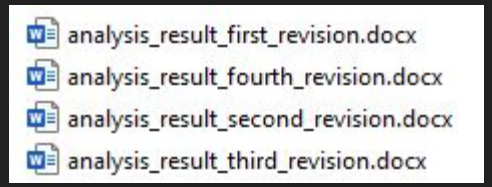
But the truth is

```
3703
3704  # Get all the CpG sites used in the analysis to form the background
3705  all <- DMPs$Name
3706  # Total number of CpG sites tested
3707  length(all)
3708
3709  par(mfrow=c(1,1))
3710  gst <- gometh(sig.cpg=sigCpGs, all.cpg=all, plot.bias=TRUE)
3711
3712  # Top 10 GO categories
3713  topGSA(gst, number=10)
3714  gst[,gst$ONTOLOGY == "BP"]
```

It will be hard to track the changes you made and to go back to that point if you’re not using any version control system (VCS)

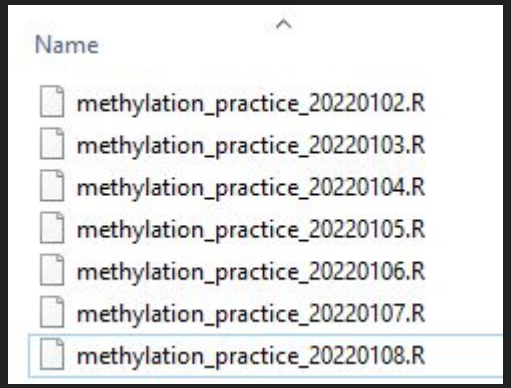
Well we have the conventional way of doing that

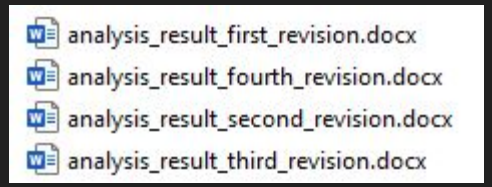




We can create multiple files and put a date suffix at the end of it’s name right?

Well we have the conventional way of doing that





But this one is not practical! Why? Not only it increases the file size but it will also make the project directory less tidier

Second Scenario

Collaboration

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

targets <- read.metharray.sheet(base_dir, pattern="SampleSheet.csv")

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

Mbak Dwi's task

```r
par(mfrow=c(1,2))
barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)], las=2,
        cex.names=0.8, ylab="Mean detection p-values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)), fill=pal,
       bg="white")

barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)], las=2,
        cex.names=0.8, ylim=c(0,0.002), ylab="Mean detection p-values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)), fill=pal,
       bg="white")
```

Mbak Zahra's task

Sometimes the project is big enough that it requires collaboration

Collaboration

```r
base_dir <- system.file("extdata", package = "minfiData")
base_dir <- "recfon_practice/data"

ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

targets <- read.metharray.sheet(base_dir, pattern="SampleSheet.csv")

# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet

# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,sep=".")
sampleNames(rgSet) <- targets$ID
```

Mbak Dwi's task

```r
par(mfrow=c(1,2))
barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)], las=2,
        cex.names=0.8, ylab="Mean detection p-values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)), fill=pal,
       bg="white")

barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)], las=2,
        cex.names=0.8, ylim=c(0,0.002), ylab="Mean detection p-values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)), fill=pal,
       bg="white")
```

Mbak Zahra's task

Not only codes but all of the things that are related to the project, like report writing etc

Version Control System solves this problem

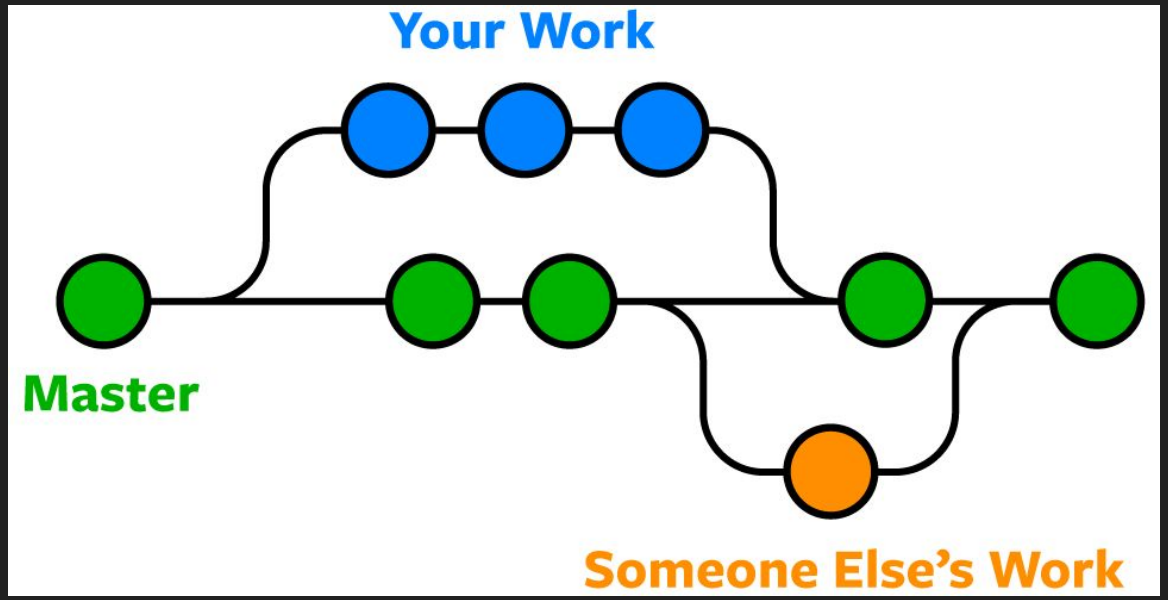


Image credit https://www.nobledesktop.com/learn/git/what-is-git

So how do we overcome such problems and doing the best practice instead of the traditional way?

Yes we have this thing called Git

What is Git

- Git is a Version Control System
  - Helps handling changes and maintaining history of our projects
  - Not only that it also has more features like **branch** and **merge**
- Git helps us revert back to the older version of our work in case something didn’t go well
- Git provides easier access for collaboration
- Git has been widely used by bioinformaticians, software developers and etc.

Installation

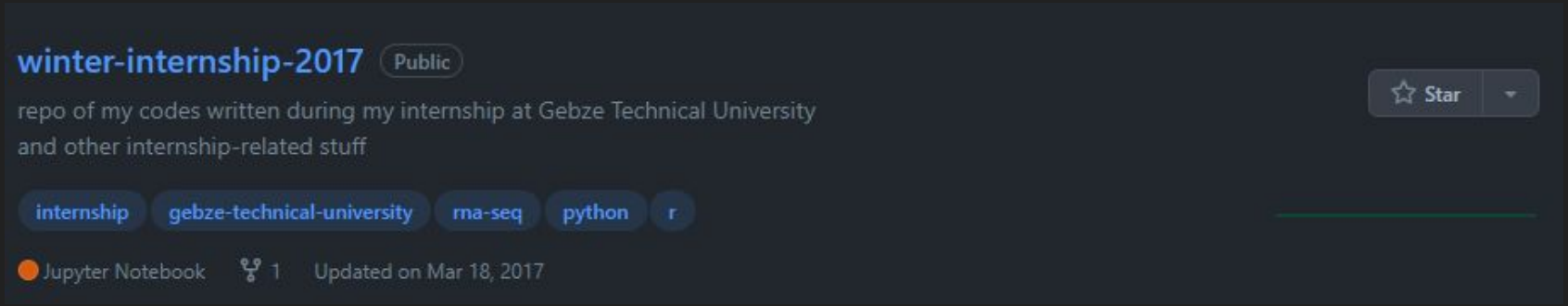
- There are multiple ways on how to install git
- The easier way when we're working with Windows computer is to download it from https://git-scm.com/downloads
- If you're using UNIX based system you should follow this instruction https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

GitHub account

- For this session we will be using GitHub to store our codes and analysis results
- To create an account go to github.com and register it with your email
- There are other providers for Git cloud but for now we will focus on using Github, they use the same command to so no worries with the commands

Repository

- A folder, usually used to organize a single project
- Repositories can contain folders and files, images, videos, spreadsheets, and data sets -- anything your project needs.
- Often, repositories include a README file, a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository.



Let’s start experimenting with Git!

Basic Git Commands

git add

git commit

git push

Basic Git Commands

Add information to the staging area

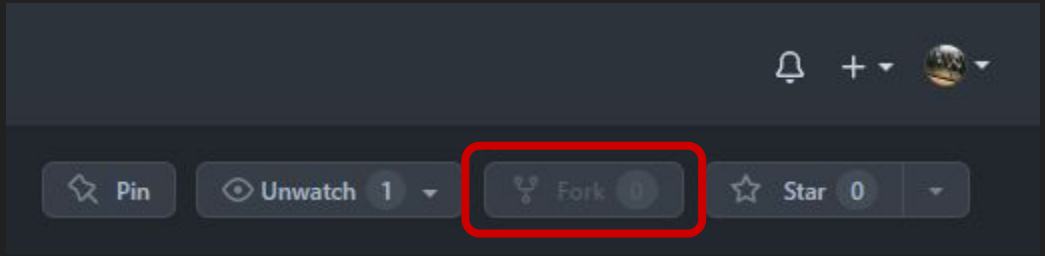
Bundle all of the file in staging area and create commit

Export the updated changes to remote repository

Git for Collaboration

git fork

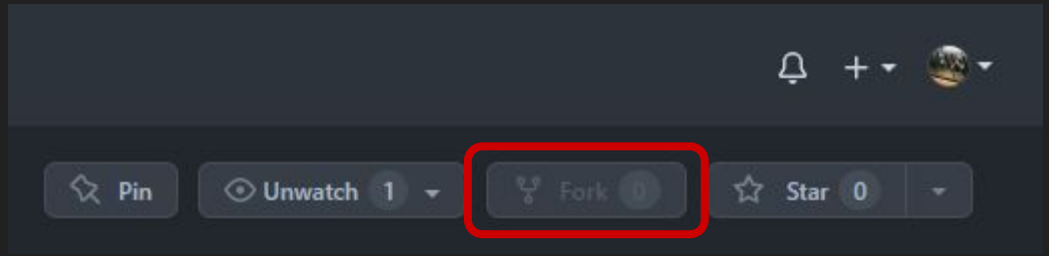
- Fork == Copy
- Forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea
- You can fork a repository to create a copy of the repository and make changes without affecting the upstream repository



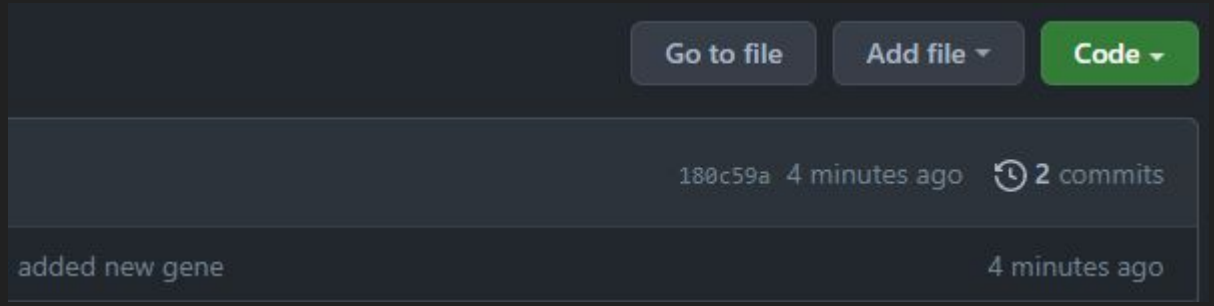
https://docs.github.com/en/get-started/quickstart/fork-a-repo

How to fork a repository

1. Choose the repository to collaborate

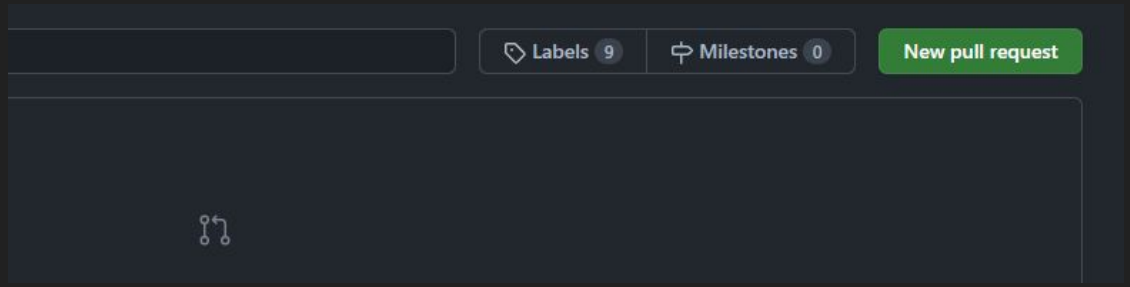


2. Clone the forked repository to your local computer



How to submit update to main repository

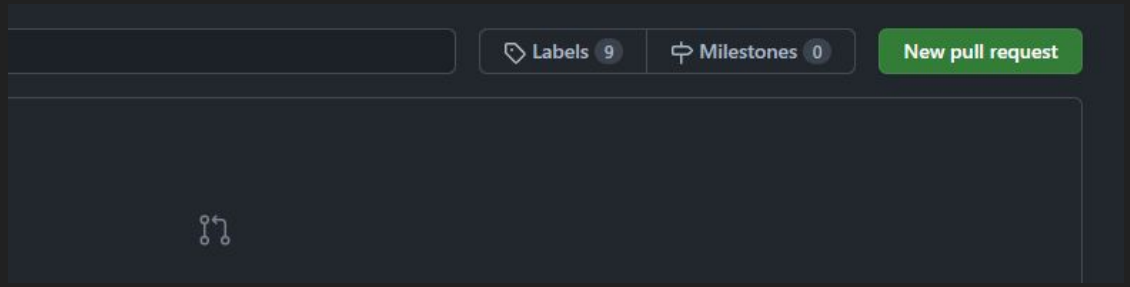
- **You have to create something called pull request**



Pull request is basically like a ticket that you send to the maintainer of the project, saying that hey I made a change to the project and I want my changes to be applied to the project

How to submit update to main repository

- **You have to create something called pull request**



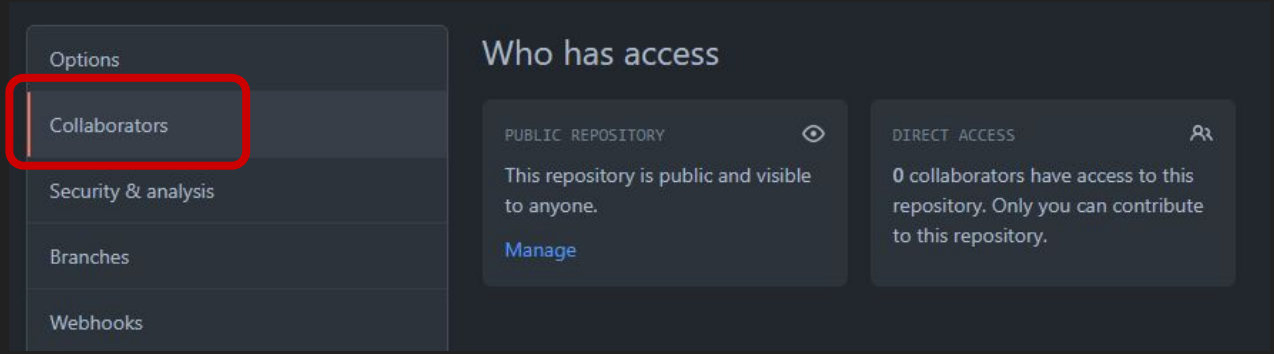
The maintainer of the project then will review your work (the changes that you made) and if if the maintainer thinks it’s ok, the maintainer will merge your project

Let’s practice

1. Fork the repository from https://github.com/hariesramdhani/test_repository
2. Your task now is to add new data of gene of your choice go to NCBI nucleotide search and get the FASTA file for it
3. Save your file in the folder called /data inside the repository
4. Push your changes to your remote repository
5. Create a pull request to the main repository https://github.com/hariesramdhani/test_repository

git collaboration

- These days git services made it easier for people to collaborate!
- People will work in the same repository but using different branch (best practice)
- You just have to invite your team member so they can work together on the same repo



Best practice when collaborating

- Update your local repo with **git pull origin main**
- Make your changes and stage them with **git add**
- commit your changes with **git commit** -m
- upload the changes to GitHub with **git push origin main**

git branch

- Sometimes during a project multiple people work on multiple small task
- Branching allows us to create a new independent line of work using the same repository without affecting the main project
- For example