

GLT tutorial

Pauline Hubert and Nadia Lafrenière

FPSAC software days, July 2019

Motivations

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRADSCHOOL?????.doc

JORISS CHAN © 2012

What software to use? Pros and cons

| | Pros | Cons |
|----------|---|---|
| Email | "Easy" to use | So many! |
| Overleaf | Many people can work at the same time | Only online, latex only |
| Dropbox | Easy to use | No two people can work at the same time, no version control |
| Git | Version control, automatic fusion of modifications, any file type | More difficult to start |

What is Git?

Git is a *version control software*.

- ▶ It keeps track of every modifications.
- ▶ You can go back in time to a previous version.
- ▶ It allows many people to work in parallel.

Online servers for Git

Store your data somewhere.

- ▶ Local: on your computer.
- ▶ Online servers: Github, Bitbucket, GitLab, your university server, etc.

If you use an online server, you will also have a local version on your computer. If you want to work with other people, you have to use a server.

Now that you are convinced... Installation

- ▶ Linux: Type in shell `sudo apt install git`
- ▶ MacOS: Go to <https://git-scm.com/download/mac>
- ▶ Windows: Go to <https://gitforwindows.org>

To share code, you need to identify yourself:

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email "john.doe@email.com"
```

Please, do not use this example as is, put your real name instead!

To get code from a remote repository:

```
$ git clone https://address-of-the-repo.git
```

Maybe some of you did, yesterday,

```
nadia@xps:~/software$ git clone git://github.com/sagemath/sage.git
Cloning into 'sage'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
Receiving objects: 4% (25444/636096), 4.81 MiB | 3.20 MiB/s
```


To get code from a remote repository:

```
$ git clone https://address-of-the-repo.git
```

Maybe some of you did, yesterday,

```
nadia@xps:~/software$ git clone git://github.com/sagemath/sage.git
Cloning into 'sage'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
Receiving objects: 4% (25444/636096), 4.81 MiB | 3.20 MiB/s
```

Exercise

Clone the repository containing this talk.

`https://github.com/phubert/git_sagedays2019.git`

What's happening?

status

Lost? You wanna check if everything went well? This command is your friend!

```
$ git status
```

```
nadia@xps:~/Documents/Exposés mathématiques/2019/git_sagedays2019$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   git-tutorial.tex

no changes added to commit (use "git add" and/or "git commit -a")
```

First steps: Starting to use Git on a project

`init`

Two ways to start:

- ▶ Use an existing local project.
- ▶ Create a new project on a server (e.g. Github) and clone it.

If you already have a project.

1. In the directory containing your files, run the command
`git init`.
2. (optional, recommended) Then, you can put your files on the server. For that, create a new *empty* repository on the server and follow the instructions.

The commands you will need are

```
$ git remote add origin REPO_URL
```

```
$ git push -u origin master
```

Once you worked on your code/text, you will want to save the modifications.

To select the (new or modified) files that will be added to the modification use the command

```
$ git add FILE_NAME1 FILE_NAME2 ...
```

To save modifications, use the command `git commit`.

Each time you make a commit, a text editor appears in your terminal and you have to write a short (but *meaningful*) comment to describe your modifications.

If you modified only files that were already in the git repository you can also list the files directly in the commit command.

```
$ git commit FILE_NAME1 FILE_NAME2 ...
```

Two usefull options : `git commit -OPTS`, where `-OPTS` is
`-a` allows you to save all the modified files with the same commit message.

`-m "Comment"` allows you to directly write your commit message in the command.

For the moment, all your modifications are only saved in your local copy of your project.

After committing, you may want to share your modifications by adding them to the online repository. The command for that is

```
$ git push
```

You may be asked to enter your username and password at this point.

You don't have to push every time you commit. You can push several commits at once, but your collaborators will not see the changes until you push.

Getting updated code & solving conflicts

pull

To obtain the updated version of the project, use

```
$ git pull
```

"What if one of my collaborator worked at the same time as me?"

If this happens, you will get an error message when you try to push.
Then, read and follow the instructions!

You can look at what other people did:

► on a software

```
diff --git a/src/sage/dynamics/arithmetic_dynamics/projective_ds.py b/src/sage/dynamics/arithmetic_dynamics/projective_ds.py
index f227882..179468a 100644
--- a/src/sage/dynamics/arithmetic_dynamics/projective_ds.py
+++ b/src/sage/dynamics/arithmetic_dynamics/projective_ds.py
@@ -53,7 +53,7 @@ AUTHORS:
# *****
from __future__ import print_function, absolute_import

-from sage.arith.misc import is_prime
+from sage.arith.misc import (is_prime, is_square)
from sage.categories.fields import Fields
from sage.categories.function_fields import FunctionFields
from sage.categories.number_fields import NumberFields
@@ -3408,7 +3408,7 @@ class DynamicalSystem_projective(SchemeMorphism_polynomial_projective_space,
                                break
                                return points
                                else:
-                                    raise NotImplementedError("ring must a number field or finite field")
+                                    raise NotImplementedError("ring must be a number field or finite field")
+                                else: #a higher dimensional scheme
                                    raise TypeError("use return_scheme=True")
                                else:
@@ -5199,9 +5199,11 @@ class DynamicalSystem_projective_field(DynamicalSystem_projective,
                                pass
                                return Conj

-    def is_conjugate(self, other):
+    def is_conjugate(self, other, R=None):
+        r"""
-        Return whether or not two dynamical systems are conjugate.
+        Return whether two dynamical systems are conjugate over their
+        base ring (by default) or over the ring R entered as an
+        optional parameter.
```


You can look at what other people did:

► on your files

```

Soit  $G$  un groupe fini et soit  $S$  un  $G$ -module simple. Soit  $\chi$  le caractère de  $S$ . Alors, on
note  $\rho_S$  l'élément de  $\mathbb{C}G$ 
-    $\rho_S = \frac{1}{\dim(S)} \sum_{g \in G} \overline{\chi(g)} g$ 
+    $\rho_S = \frac{1}{\dim(S)} \sum_{g \in G} \overline{\chi(g)} g$ 
+   où  $\overline{\chi}$  désigne le conjugué complexe de  $\chi$ .
La multiplication à droite par cet élément constitue la fonction sur l'algèbre du groupe  $G$ 
\begin{align*}
\text{isoproj}_S : \mathbb{C}G &\rightarrow \mathbb{C}G^{\{S\}} \\
&w \mapsto w \cdot \rho_S, \\
&w \mapsto w \cdot \rho_S.
\end{align*}
-   où  $\overline{\chi}$  désigne le conjugué complexe de  $\chi$ .

```

Reviewing changes

diff

To know what has been change since last commit:

```
$ git diff
```

or since commit1

```
$ git diff commit1
```


or between commit1 and commit2

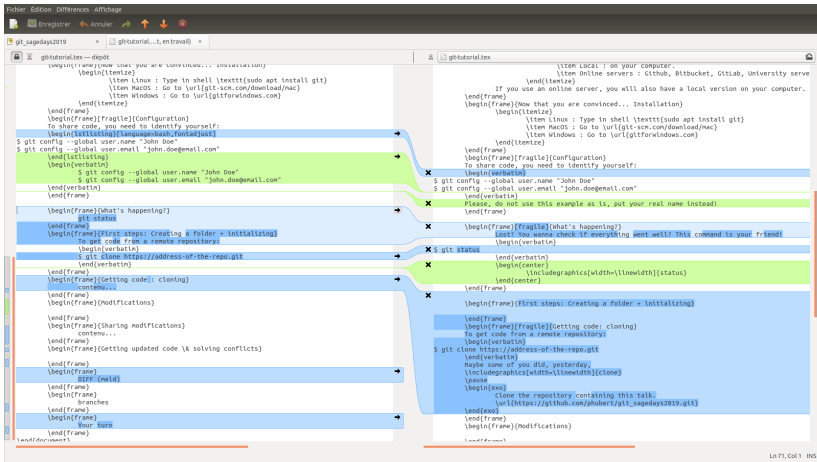
```
$ git diff commit1 commit2
```

To recover the "name" of commit1, you must type `git log` (this is the history of the repository). The name is very long! (e.g.: `df14e25763073c18e30ebeede3d34ef466dd08c8`)

Reviewing changes

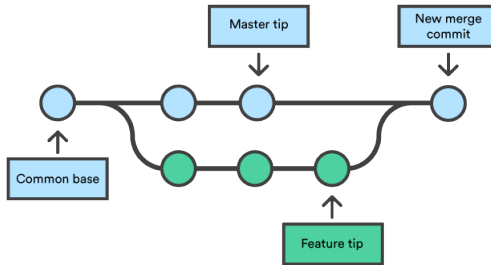
diff

You can do it with a graphical user interface (e.g. Meld ) .



Branches

Image from Bitbucket web site



Create a new branch : `$ git branch BRANCH_NAME`

Navigate between branches : `$ git checkout BRANCH_NAME`

Merge branches : `$ git merge BRANCH_NAME`

Conclusion/Moral

PLEASE, PLEASE, PLEASE! Read the error messages and outputs.

Conclusion/Moral

PLEASE, PLEASE, PLEASE! Read the error messages and outputs.

Some references



An awesome tutorial:

swcarpentry.github.io/git-novice/



More tutorials, by Github and BitBucket



Some cheatsheets, by Software Carpentry and Github



Bitbucket, Github and Gitlab all have extended documentation on their website

Your turn!

Two-by-two, try the following:

- ▶ Go to the Exercise folder of our repository (you *need* to clone the repository from `https://github.com/phubert/git_sagedays2019.git`).
- ▶ Create your own branch (name it after the concatenation of your names).
- ▶ Edit three of the files.
- ▶ Send your files to the server. *Don't forget to put a relevant commit message!*
- ▶ We will add mistakes in your files.
- ▶ Get the modifications and correct the mistakes. Send everything back to us (not by email, of course...)