# GitHub Practicals for the 21.10.2021

October 21, 2021

## 0.1 Recall:

Here, I recall what we did during our last GitHub practical session.

### 0.1.1 From our previous GitHub practical, we did the the following:

Step 1 : Initialize your repository. Navigate to `/home`

```
$ cd
$ mkdir simplestats
$ cd simplestats
$ git init
Initialized empty Git repository in /home/me/simplestats/.git/
```

Step 2 : Browse the directory's hidden files to see what happened here. Open directories, browse file contents. Learn what you can in a minute.

```
$ ls -A
.git
$ cd .git
$ ls -A
HEAD        config      description hooks       info        objects     refs      branches
```

Step 3 : Use what you've learned. You may have noticed the file called description. You can describe your repository by opening the description file and replacing the text with a name for the repository. We will be creating a module with some simple statistical methods, so mine will be called "Some simple methods for statistical analysis". You may call yours anything you like.

```
$ gedit description
```

Add a File to Your Local Repository

Step 1 : Create a file to add to your repository.

```
$ touch README.md
```

Step 2: Verify that git has seen the ne file.

```
$ git status
# On branch master

# No commits yet

# Untracked files:
```

```
  #(use "git add <file>..." to include in what will be committed)
    README.md

# nothing added to commit but untracked files present (use "git add" to track)
```

Step 3 : Inform git that you would like to keep track of future changes in this file.

```
$ git add README.md
```

### 0.1.2 git status : Checking the Status of Your Local Copy

The files you've created on your machine are your local "working" copy. The changes your make in this local copy aren't backed up online automatically. Until you commit them, the changes you make are local changes. When you change anything, your set of files becomes different from the files in the official repository copy. To find out what's different about them in the terminal, try:

```
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README.md
#
```

If you have not done it, do it now before you continue with the instructions below.

## 0.2 git branch : Listing, Creating, and Deleting Branches

Branches are parallel instances of a repository that can be edited and version controlled in parallel. They are useful for pursuing various implementations experimentally or maintaining a stable core while developing separate sections of a code base.

Without an argument, the **branch** command lists the branches that exist in your repository.

```
$ git branch
* master
```

The master branch is created when the repository is initialized. With an argument, the **branch** command creates a new branch with the given name.

```
$ git branch experimental


$ git branch
* master
  experimental
```

To delete a branch, use the `-d` flag.

```
$ git branch -d experimental
```

```
$ git branch
* master
```

## 0.3 git checkout : Switching Between Branches, Abandoning Local Changes

The **git checkout** command allows context switching between branches as well as abandoning local changes.

To switch between branches, try

```
$ git branch add_stats

$ git checkout add_stats

$ git branch
```

How can you tell we've switched between branches? When we used the branch command before there was an asterisk next to the master branch. That's because the asterisk indicates which branch you're currently in.

---

### 0.3.1 Exercise : Copy files into your repo

Make sure we have a **good copy** of `stats.py` and `test_stats.py` in the directory `/simplestats` .

Now let's add them to our repo, but in the current branch.

```
$ git add *stats.py

$ git commit -m "Adding a first version of the files for mean."
```

---

### 0.3.2 Exercise : Add a comment to one of the stats files.

1. Open either `stats.py` or `test_stats.py` in the text editor of your choice.
2. Add a comment line to this file with your name. Comment lines in Python start with the `#` character. For exmaple,

   ```
   # AIMS Senegal: Hard time with Git.
   ```

3. Commit the changed files to your repo.

---

## 0.4 git merge : Merging Branches

At some point, the `add_stats` branch may be ready to become part of the `master` branch. In real life, we might do a lot more testing and development. For now, let's assume that our mean function is ready and merge this back to the master. One method for combining the changes in two parallel branches is the **merge** command.

```
$ git checkout master

$ git merge add_stats
```