GIT Course - UFG Goiania - 25, 26 Febereiro 2014
Day 1 - Practice Script

## Practice 1.1

### **Objectives:**

- Create a repository: init
- Add and track changes to the repository: add, commit
- Remove files: add -u
- See repository changes over time: log
- Go back and forward in time
- Ignoring files with .gitignore
- Remove some change: reset

### Story:

- Simulation of the process of publishing a paper in a journal
- Writing the paper, making changes and tracking these changes with git

### Start!:

- First day: We start writing the paper
    - create a folder for the paper (eg: practices/1.1) and a file called "article.txt" with text "first day"

- Turn this folder into a git repository
    - > git init
    - Note that:
        - a new hidden folder '.git' was created
        - we can type now 'git status': 'article.txt' is 'untracked'

- Say git to start 'tracking' this file: add + commit (c1)
    - > git add article.txt
    - Type 'git status': article.txt shows as in the list 'changes to be committed'
    - > git commit -m "a hard first day of work"

- Second day: write the 'Introduction' chapter of the article (c2)
    - edit article.txt and add a title '1. Introduction' followed by 10 lines (important) of dummy text. Remove text from first day.
    - > git add article.txt
    - > git commit
    - this time we don't add the comment in the command so git shows us a text editor for writing it. write this comment: "write the introduction" and exit the editor

- Third day: write the 'Methods' chapter (c3)
    - edit article.txt and add a title '2. Methods' followed by some dummy text.
    - > git add article.txt
    - > git commit -m "write the methods"

- Review evolution of repository (git log)
    - > git log -3

- Fourth day: write the 'Analysis' chapter and adds a 'plots' file (c4)
    - edit article.txt and add a title '3. Analysis' followed by some dummy text.
    - create a file called 'plots.txt' with some 'plot 1... plot 2' text
    - > git status
        - Note that 'article.txt' is 'modified' and 'plot.txt' is 'untracked'
        - see http://git-scm.com/book/en/Git-Basics-Recording-Changes-to-the-Repository for explanation on file states
    - > git add .
        - This time we 'stage' all the changes in the folder with 'git add .'. Add everthing (not ignored) to the 'staging' area
        - Alternative to 'git add article.txt plot.txt'
    - > git commit -m "write analysis and adds plots"

- Fifth day: change of mind, remove the plots (c5)
    - remove file 'plots.txt' from folder
    - > git status
    - plots.txt shows as 'deleted'
    - > git add -u .
    - '-u' needed to 'update the index' and track the 'deletion' of file
        - Note: more info on any command typing --help. eg: > git add --help
    - > git commit -m "remove the plots"

- Sixth day: writes 'Conclusion' and ignore a nasty file (c6)
    - edit article.txt and add a title '4. Conslusion' followed by some dummy text.
    - imagine we have edited the file with a software that creates some sort of 'configuration file' in our folder called 'nasty_file.ini'. eg: '.Rhistory' when working with RStudio
        - create a file called 'nasty_file.ini' with some dummy text
    - > git status
    - the file is 'untracked', if we make 'git add .' it will be added to the 'staged changes'
    - create a file called '.gitignore' with the content: 'nasty_file.ini'
    - > git status : doesn't show nasty_file.ini any more
    - commit .gitignore to the repo with message "write conclusions and ignore nasty file"

- Exercise on .gitignore
    - create a folder 'vendor' with two files 'lib1.R' and 'lib2.R'
    - ignore this: use 'vendor' in .gitignore

- Seventh day: you need to pick up one of the removed plots!! (c7)
    - we need to go back in time and pick what we removed

    - Method 1 (dirty): go back, copy and paste
        - > git log -5
        - find the 'hash' of the commit in which plots where added:
        - go back in time
            - > git checkout 763abc
        - we are out of the 'master' branch
        - open 'plots.txt' and copy some of the content to some temporary file (wherever)
        - go forward in time (back to 'master' branch)
            - > git checkout master
        - create a file 'plots.txt' and paste the contents
        - commit changes

    - Learn to undo. But we don't like this method. Remove last commit (forever)
        - see commit hash of the previous commit "write conclusions and ignore nasty file"
        - > git log -2

- > git reset COMMITHASH
        - > git log -2 (again in c6)
        - remove plots.txt

    - Method 2 (nice): checkout some past file
        - git checkout COMMITHASH -- plots.txt
        - edit plots.txt remove what you don't want and commit changes with message "plot 2 back in time"

- Eight day: move plot to 'Analysis' chapter
    - apply learned stuff
        - make a commit with the 'plot 2' from plots.txt is in 'Analysis' chapter and remove plots.txt
    - commit with message "ready for peer review"

------------------ end of practice 1 ------------------------------

## Practice 1.2

### Objectives

- Create and move through branches
- Merge branches
- Fix merge conficts

### Story

Now the article is ready we send it to a journal. We will track the particular modifications they ask us using git 'banches'

### Start

- Read 'Journal of Good Science' reply: rejected

- First day: prepare the paper for 'Journal of Better Sciene'

    - what is a branch? --> explain
    - create a 'branch' for the adapted version of the paper for this particular journal
    - > git branch
        - shows the branches we have: master
    - > git checkout -b jobs
        - crates and moves to a new branch called 'jobs' for the journal
        - any commit will go to this branch (not to master)

    - Adapt paper to J.O.B.S. policies:
        - Introduction must have just 2 lines
        - Must come with a file 'data.txt' with 'Data Anexes'
        - make changes and commit (in branch jobs) as "adaptation to jobs journal policy"

    - > git log -3

- Second day: Read 'Journal of Better Science' reply: rejected

    - Also they point out that you have a typo in line 3 of 'Analysis'
    - go back to master branch, fix the typo and commit it (in master)
        - > git checkout master

- Third day: prepare the paper for 'Journal of Awesome Science'

    - create a branch for 'joas' based on branch 'master'
        - > git checkout master
            - back to 'master' branch
        - > git checkout - b joas
        - > git branch
            - see all three

    - Adapt paper to J.O.A.S. policies:
        - 4 lines introduction. Must rewrite lines. Different text
        - Add chapter 5. Thanks to
        - Must add a file with the cvs of the authors called 'authors.txt'.
        - Add it and commit (in joas)

        - Must add a file with the data set: data.txt created for 'jobs'
            - Need to merge with changes in 'jobs'
            - > git merge jobs
            - !!! CONFLICT: becouse the same lines in article.txt are modified in the other paralell
version
            - note that file 'data.txt' is in place

        - Fix confilct
            - edit article.txt
            - note how conflict is shown in the introduction
            - note there is no conclict in '5. Thanks to'
            - edit and commit as "adaptation to joas journal policy"

    - Send to JOAS ....

- Fourth day: Read 'Journal of Awesome Science' reply: ACCEPTED!

    - Apply all changes in joas branch to master branch
        - > git checkout master
        - > git merge joas
            - messages says "Fast-forward". 'master' is not merging just moving on in time
        - (optional) remove branch 'joas' (now it is merged)
            - > git branch -d joas
        - (optional) remove branch 'jobs'
            - > git branch -d jobs

------------------- end of practice 2 ---