Introduction au Bootcamp
Contrôle de Version, Git

W1D1

Jean-Marc Marty - 03/06/20

Présentation Katya

Présentation de JM et Marwan

Ice breaker :) 

Introduction au Bootcamp

Introduction au Bootcamp

But

- Dans **6 mois**, vous serez capable de :
  - *Récupérer* de la donnée
  - *Préparer* un jeu de données en vue d'une analyse
  - *Analyser* la donnée en vue de répondre à des **questions** que l'on se pose
  - *Restituer* les résultats des analyses sous la forme de graphes
  - *Construire* des modèles prédictifs
  - *Trouver* des clusters dans un jeu de données

Introduction au Bootcamp

Concrètement…

- *But :* Dans **6 mois**, vous serez capable de :
  - *Récupérer* de la donnée : Relational Database/MySQL, Web Scraping, APIs
  - *Préparer* un jeu de données en vue d'une analyse : Pandas, Data Wrangling/Cleaning
  - *Analyser* la donnée en vue de répondre à des **questions** que l'on se pose : Descriptive Statistics, Probability Distributions, Significance Testing, Discriminant, Factor, Cluster and Regression Analysis
  - *Restituer* les résultats des analyses sous la forme de graphes : Tableau, Matplotlib and Seaborn Visualization libraries
  - *Construire* des modèles prédictifs: Feature Extraction and Engineering, Supervised Learning, Model Evaluation, Hyperparameter Tuning
  - *Trouver* des clusters dans un jeu de données: Clustering

Conseils

Conseils

- Avant chaque cours :
  - Prenez un peu de temps pour essayer de synthétiser pour vous le cours précédent ou les cours précédents (une piqûre de rappel au bout de quelques mois peut être très bénéfique).
  - Lisez le contenu du cours suivant pour vous préparer mentalement, prenez de l'avance.
- Pendant le cours :
  - Ecoutez bien et **posez des questions** pendant la partie théorique si besoin.
  - Essayez de faire en même temps que moi les exemples applicatifs.
- Pendant le Lab :
  - Evitez de copier/coller le contenu du cours, retapez les lignes pour utiliser votre mémoire physique.
- Ne multipliez pas trop les sources de contenu, chacun a sa méthodologie et dans un premier temps il est mieux de se restreindre.
- Si besoin, sources éventuelles :
  - *Python for Data Analysis (*https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf)
  - *Hands-on Machine Learning w/ Scikit-Learn* (https://www.lpsm.paris/pageperso/has/source/Hand-on-ML.pdf)

🔥 C’est parti ! 🔥

Utilisation du terminal

Mac / Windows

Utilisation Terminal sur Mac

Commandes principales à connaître

- `cd <directory> / cd ..`
- `ls`
- `mkdir <directory>`
- `cat`
- `head <file>`
- `rm <file>`
- `mv <file-old> <file-new>`
- `cp <file> <directory>`
- Cheat Sheet: https://www.git-tower.com/blog/command-line-cheat-sheet/

Command Prompt sur Windows

- Sinon : https://cdn.makeuseof.com/wp-content/uploads/2017/02/Essentials-Windows-CMD-Command-You-Should-Know-2.pdf

- *Mon conseil :* Ayez un Mac (ou installez Ubuntu) :) En informatique, essayez toujours de vous ramener à des situations “classiques”. La plupart des Data Analysts sont sur Mac ou Linux, donc la plupart des librairies / logiciels vont bien fonctionner sur Mac ou Linux. En général sur Windows on a beaucoup plus de problèmes.

Contrôle de version et Git

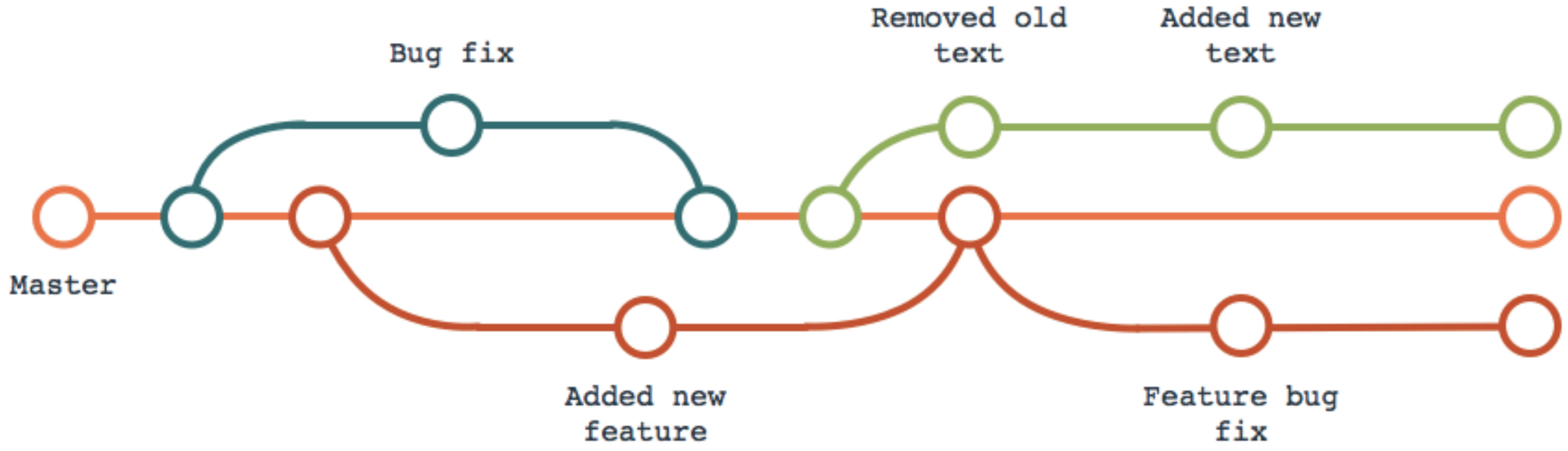
Contrôle de Version

Contrôle de version = Système de sauvegarde sur une codebase

- But :
  - Sauvegarder les différentes étapes de son code pour pouvoir revenir en arrière.
  - Travailler en parallèle sur une autre version.
  - Sauvegarder son code sur le Cloud en cas de perte/vol/casse/suppression de ses données.
  - Travailler à 10 sur le même code : les versions vont être démultipliées, il faut un système robuste qui permet de ne rien perdre/écraser.
- Pour tous ces cas de figure, on va vouloir utiliser *un logiciel de contrôle de version*. Exemple: **git**.

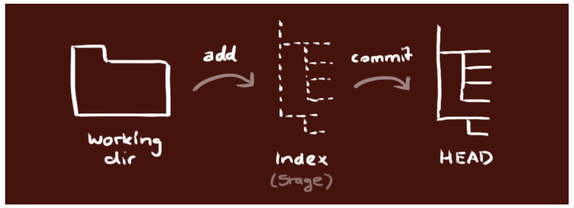
Git : Un peu de vocabulaire

- *Repo/Repository* : Un dossier de travail + l'historique de travail Git.
- *Commit :* Une sauvegarde.
- *Branch :* Une version parallèle du code.
- *Merge* : Fusionner des versions parallèles du même code. Attention aux conflits !
- *Push :* Uploader dans le cloud nos modifications.
- *Pull :* Downloader les modifications du cloud en local.
- *Clone :* Downloader un repo et créer un lien encore le repo local et le repo remote.
- *Github :* un service d'hébergement Cloud dédié à Git. Concurrents : Gitlab, Bitbucket.
- *Pull Request* : Demander à l'administrateur d'un repo de merger ses modifs sur une branche.

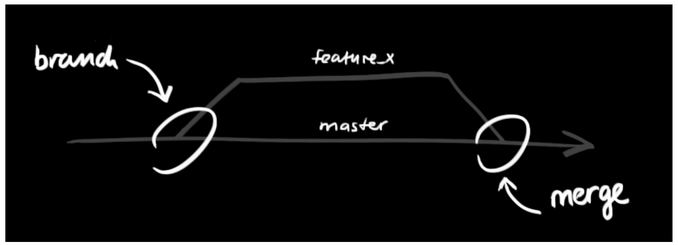

Bug fix
Removed old text
Added new text
Master
Added new feature
Feature bug fix

Git / Github Workflow

- **Initialisation :**
  - Soit on part d'un dossier local :
    - git init
  - Soit on part d'un repo remote :
    - git clone
- **Connecter un repo local avec un repo remote (pas nécessaire si on a fait un git clone)**
  - git remote add origin <server>
- **Créer un nouveau commit :**
  - git add * / git add <filename>
  - git commit
- **Pusher les modifications en remote :**
  - git push origin master / git push
- **Puller des modifications du remote :**
  - git pull
- **Créer une nouvelle branche :**
  - git checkout -b <branch_name>



- **Fusionner une branche avec la branche active :**
  - git merge <branch>



- **Cheat Sheet: https://raw.githubusercontent.com/hbons/git-cheat-sheet/master/preview.png**