Une boite à outils techniques pour le data scientist

Cédric Bohnert

23/10/2021

Utiliser la CLI (interface de ligne de commande) Linux

Connaitre les bases de l'utilisation de la ligne de commande est nécessaire au data scientist pour installer des programmes et pour gérer une structure de fichiers.

Quelques commandes pour installer des programmes :

- `sudo` = confie les droits d'execution, de lecture et d'écriture à l'utilisateur courant (demande son mot de passe administrateur)
- `apt-get install` = télécharge et installe un package d'un dépot distant
- `apt-get update` = met à jour tous les dépots
- `apt-get upgrade` = met à jour tous les packages de l'utilisateur courant

Exemple : installer le système R au complet :

- `sudo apt-get update`
- `sudo apt-get install r-base`

Quelques commandes de gestion de fichiers à connaitre :

- `/` = dossier racine
- `~` = dossier utilisateur courant
- `pwd` = affiche le nom du dossier courant
- `clear` = efface l'affichage de la console
- `ls` = liste les éléments du répertoire courant
  - `-a` = voir tous les éléments, y compris cachés
  - `-l` = ajoute les détails
- `cd` = change de dossier
- `mkdir` = crée un nouveau dossier
- `touch` = crée un fichier vide
- `cp` = copie un fichier
  - `cp <file> <directory>` = copie un fichier dans un dossier
  - `cp -r <directory> <newDirectory>` = copie tous les éléments d'un dossier dans un nouveau dossier
    * `-r` = recursive (copie récursive)
- `rm` = efface un fichier
  - `-r` = efface un dossier et tout son contenu
- `mv` = déplace un fichier
  - `move <file> <directory>` = déplace un fichier dans un dossier
  - `move <fileName> <newName>` = renomme le fichier
- `echo` = affiche l'argument/les variables données
- `date` = affiche la date courante

Quelques exemples :

- Création d'un nouveau script R dans un dossier spécifique :
  - `cd ~/Datascience/Projet1/`
  - `touch mon_script.R`
- Copie d'un fichier téléchargé dans un dossier utilisateur nouvellement créé :
  - `mkdir ~/Documents/Projet`
  - `cp ~/Téléchargements/mon_fichier_téléchargé ~/Documents/Projet/`
- Suppression du dossier Téléchargements_temporaires et de tous ses éléments :
  - `rm -r ~/Téléchargements_temporaires`
- Affichage de du nom d'utilisateur courant :
  - `echo $USER`
- Affichage des dossiers bibliothèques R installées dans le système (dépend du système) :
  - `ls -la R/x86_64-pc-linux-gnu-library/4.1`

Pour approfondir : De très très nombreuses opérations peuvent se réaliser en ligne de commande, voir les ressources ci-dessous :

- Introduction à la ligne de commande
- La ligne de commande sous Ubuntu
- Les meilleurs cours et tutoriels pour apprendre le système Linux

Utiliser git et GitHub

Git est un outil de contrôle de version et Github est une interface ainsi qu'un espace de stockage pour projets de développement très utilisé dans le monde de la data science.

Un outil de contrôle de version est un programme qui permet de travailler sur une seule copie de fichier.

Lorsqu'on fait des changements sur un fichier, le système sauvegarde l'historique des changements et permet de revenir sur une modifications.

Vocabulaire :

- Un dépot est une espace de stockage en ligne dans lequel on place les fichiers du projet
- Un 'commit' est une sauvegarde à un instant t (comme une photographie) d'un fichier ou d'un ensemble de fichiers accompagné d'un message
- Un 'push' est une mise à jour du dépot distant : les fichiers sont envoyés sur le dépot après avoir fait un commit
- Un 'pull' est une mise à jour des fichiers locaux à partir de ceux du dépot
- Une branche est une copie différente des fichiers d'un dépot
- Un 'merge' est une fusion d'une branche et le dépot principal
- Un conflit survient lorsque plusieurs personnes ont édités un fichier et souhaitent le fusionner avec le dépôt principal
- un clone est une copie locale d'un dépot : lorsqu'on on commence à travailler sur un projet existant, on commence par cloner le dépot
- un 'fork' est une copie personnelle d'un projet d'une autre personne

Bonne pratiques :

- Faire un seul commit par problème rencontré dans le projet
- Bien renseigner les commit
- Puller et pusher souvent

Séquence de travail :

1. Effectuer des modifications dans l'espace de travail (script, markdown, etc)
2. Mettre à jour / sauvegarder les modifications
3. Effectuer un 'commit' dans le dossier local
4. Effectuer un 'push' dans le dépot distant

Commandes courantes :

- `git add .` = ajoute tous les nouveaux fichiers à suivre
- `git add -u` = met à jour les suivivs pour les fichiers renommés ou effacés
- `git add -A` = réalise les deux opérations précédentes
  - notez que `add` est réalisé avant tout 'commit'
- `git commit -m "message"` = 'commit' les modifications que vous voulez sauvegarder dans la copie locale
- `git checkout -b branchname` = crée une nouvelle branche
- `git branch` = informe l'utilisateur sur la branche de travail courante
- `git checkout master` = retourne sur la branche maitresse
- `git pull` = fusionne les modifications dans les branches/dépot de l'utilisateur propriétaire
- `git push` = 'commit' les modifications locales dans le dépot distant (GitHub)

Pour approfondir :

- git - un petit guide
- git Pro - un livre

On pourrait élaborer un TP pour appliquer ces compétences en créant par exemple une structure de projet concernant :

- L'organisation de l'étude de ce cours (prise de notes, resssources, scripts, visualisations, etc.)
- Un template d'un projet type de data science (liste de tâches, liste de questions, scripts, jeux de données, visualisations, etc.)

et la stocker dans un dépot GitHub.

Utiliser Markdown

- `##` = signifies secondary heading (bold big font)
- `###` = signifies tertiary heading (slightly smaller font than secondary, not bold)
- `*` = bullet list item