

Exercice 3 : GIT & branches

Nous allons à présent voir utiliser la fonctionnalité clé de GIT que sont les « branches ». Bien que cette fonctionnalité soit également présente dans tous les systèmes de contrôle de version « modernes » (e.g., subversion), elle offre davantage de souplesse avec GIT et en constitue par conséquent un élément incontournable.

L'objectif de cet exercice sera d'apporter des modifications à un script R que vous récupérerez sur mon compte GitHub. Ce script réalise un simple graphique avec des points noirs et des points rouges, les modifications consisteront à afficher les points noirs en gris et les points rouges en orange. Pour cela, nous commencerons par créer une nouvelle branche afin de faire coexister les deux versions du code (l'initiale et celle que vous allez créer), et fusionnerons les deux branches lorsque les modifications seront réalisées (et validées).

1. **Commencez par « forker » le dépôt « tp-git_exo-branching » que vous trouverez sur mon compte GitHub.**

Pour cela :

- a. Vous rendre sur la page du dépôt : https://github.com/pmahe/tp-git_exo-branching
- b. Cliquer sur l'icône « fork » en haut à droite. Une copie du dépôt apparaîtra sur votre page GitHub : c'est « votre » version du dépôt que vous pouvez manipuler à votre guise. Notez que cette opération est valable pour tout dépôt public disponible sur GitHub.

2. **Rapatrier le dépôt en local.**

Pour cela :

- a. Aller sur votre page GitHub et copier l'URL permettant de « cloner » le dépôt.
- b. Rapatrier le code en local via la commande « `git clone` ».

3. **Exécuter le script R.**

4. **Créer une branche « change_colors » via la commande « `git branch` »** (en l'occurrence par l'appel « `git branch change_colors` »).

5. **La commande « `git branch` » permet de voir l'ensemble des branches existantes, en indiquant via un « * » la branche sur laquelle vous vous trouvez. Quelle est-elle ?**

6. **La commande « `git checkout BRANCH_ID` » permet de changer de branche (en lui spécifiant le nom de la branche sur laquelle aller).** Passer sur votre nouvelle branche et vérifier que c'est bien le cas via la commande « `git branch` ».

7. **Modifier à présent le script pour qu'il change les couleurs des points et effectuez un « `git status` »** pour vous assurer que git a bien détecté le changement.

8. **Enregistrer vos changements** (i.e., add + commit).

9. Repasser sur la branche maitre (« master ») via la commande « `git checkout` ». Le script a-t-il été modifié ?
10. On peut alors « pousser » la nouvelle branche sur le serveur distant (« remote ») via la commande « `git push` » en utilisant l'option « -u », en l'occurrence « `git push -u origin change_colors` ». Effectuer cette opération, et vérifier que vous avez à présent deux branches répertoriées sur votre dépôt GitHub.
11. Pour finaliser votre script, appliquez-lui une autre modification (e.g., changer le titre du graphique) sur votre nouvelle branche et « pousser » les modifications sur le serveur distant.
Pour cela, faire donc une combinaison « `git add` » + « `git commit` » + « `git push` », en « poussant » bien la branche « change_colors » sur le serveur distant « origin ».
12. Enfin, maintenant que vous êtes pleinement satisfaits de vos modifications, transférez les sur la branche maitre via la commande « `git merge` ».
Pour cela :
 - a. Repasser sur la branche maitre via la commande « `git checkout master` »
 - b. Fusionner la nouvelle branche via la commande « `git merge change_colors` »
 - c. Vérifier que le script a bien été modifié.
13. Rendez-vous maintenant sur votre page GitHub pour voir si le fichier distant a été mis à jour.
Ce n'est pas le cas, car les modifications ne sont pour l'instant effectives qu'en local. On peut s'en convaincre en appelant la commande « `git status` » qui nous dit que notre branche maitre (« master ») est en avance par rapport à la branche maitre distante (« origin/master »).
14. « pousser » la nouvelle version du script (sur la branche maitre) sur le dépôt distant via la commande « `git push origin master` » (et vérifier que c'est bien le cas).
15. Enfin, maintenant qu'on a terminé de modifier notre script, la branche « change_colors » n'a plus lieu d'être. On peut la supprimer via la commande « `git branch` » en utilisant l'option « -d », en l'occurrence via l'appel « `git branch -d change_colors` ». Effectuez cette opération et vérifiez que la branche a bien été supprimée via la commande « `git branch` » (sans option cette fois).
16. Cette opération ne permet néanmoins que de supprimer la branche sur votre dépôt local : si elle est effectivement supprimée en local, elle est toujours présente sur le dépôt GitHub. Pour la supprimer de GitHub, il faut utiliser la commande « `git push` » et l'option « --delete », en l'occurrence « `git push origin --delete change_colors` ». Effectuez cette opération et vérifiez que la branche a également été supprimée du dépôt GitHub.