Bonnes pratiques en R

Cette page est basée sur un document qu'Aurélien Nicosia (ULaval) a créé en 2023 appelé "Bonnes pratiques de programmation en R". Celui-ci a été mis à jour.

R étant "juste" un langage de programmation, il est techniquement possible d'utiliser un simple éditeur de texte pour écrire du R et un terminal pour lancer le code. Il est cependant bien plus commode—en particulier, pour les débutants—d'utiliser un éditeur de code comme RStudio pour lancer son code. En effet, celui-ci permet de lancer son code de façon intéractive, de pouvoir voir son environnement de travail, de tester son code, …

De plus, il vaut mieux s'assurer de travailler avec la dernière version de R et des packages dont nous avons besoin. Ainsi, nous risquons moins de rencontrer des bogues et nous pouvons profiter des dernières fonctionnalités. Un point d'attention, cependant, je vous déconseillerais de changer la version de R ou des packages une fois que vous avez commencé à travailler sur quelquechose. En effet, ce changement pourrais introduire des bogues au milieu de l'analyse (il faut donc toujours reporter les versions des packages utilisés pour la reproducibilité). Il est possible d'avoir un résumé de l'environnement de travail avec les fonctions `systemInfo()` et `packageVersion(pkg)`.

Concernant la documentation, le package `roxygen` permet de convertir les commentaires de vos fonctions et packages en une documentation propre.

Une syntaxe uniforme rend un code beaucoup plus facile à lire et à comprendre. Un guide de style énonce des normes pour avoir une syntaxe uniforme. Le guide de style du *tidyverse* est recommandé en R. Avant de présenter quelques conventions de style, notons qu'il est possible de modifier rapidement la mise en forme d'un bout de code R dans RStudio dans le menu "Code -> Reformat Selection".

Retour à la ligne et indentation

Une façon simple de rendre som code plus lisible est d'y insérer des retours à la ligne et des indentations appropriés. Par exemple, supposons que nous avons la chaîne de caractère suivante :

```
text <- "Ceci est un example"
```

Nous souhaitons corriger deux fautes dans cette phrase : le mot “example” écrit en anglais plutôt qu’en français et le point manquant à la fin de la phrase. Ceci peut se faire avec l’instruction :

```
paste0(gsub(pattern = "example", replacement = "exemple", x = text), ".")
```

Cette instruction comporte un appel de fonction imbriqué dans un autre. Elle est bien plus facile à lire comme suit :

```
paste0(
  gsub(
    pattern = "example",
    replacement = "exemple",
    x = text),
  "."
)
```

Opérateur d’assignation

En R, on utilise `<-` pour assigner une valeur à une variable et on utilise `=` pour passer des valeurs à des arguments dans un appel de fonctions.

Conventions de noms

Le guide de style du *tydiverse* préconise l’utilisation de lettres minuscules, de nombres et de l’underscore `_` pour nommer variables et fonctions. Les underscores sont utilisés pour séparer les mots dans un nom. Bien que l’on puisse trouver d’autres conventions, celles-ci sont à éviter. Dans tous les cas, il est important de choisir une convention et de la respecter. De plus, il est préférable d’éviter les accents dans les noms de variables.

Organisation du code

Lorsque le code commence à devenir long, il devient avantageux de le séparer en plusieurs fichiers. Par exemple, on peut avoir un fichier par partie de l’analyse (un pour le nettoyage des données, un pour l’analyse, un pour la visualisation, …). De plus, une analyse de données n’est généralement pas constitué uniquement de code R, e.g. fichiers de code C++, fichiers de données, fichiers du configuration, etc. Il est donc recommandé des créer des sous-dosssiers regroupant les fichiers du même type. Les projets RStudio sont parfaits pour rassembler au même endroit tous les fichiers relatifs au projet. De plus, ils permettent de faciliter le travail sur plusieurs projets simultanément en gérant le passage d’un répertoire de travail à un autre.

Quelques trucs à faire

1. Rédiger son code dans un script et l’enregistrer fréquemment. Cela permet d’éviter de perdre la trace de certaines instructions importantes parce qu’elles ont été écrites directement dans la console.

2. Il est préférable de débuter toute session de travail en R avec un environnement de travail vide. Pour ce faire, il faut désactiver la restauration automatique d'une image de session dans les paramètres. Cela permet d'être conscient de la présence des différents objets dans l'environnement de travail.
3. Ne pas utiliser la fonction `load` lorsque l'environnement de travail n'est pas vide. Cela permet de ne pas modifier un objet de l'environnement de travail en l'écrasant.
4. Ne pas utiliser la fonction `attach`. Cela permet de ne pas modifier le chemin de recherche des fichiers.
5. Sauvegarder les options et paramètres graphiques avant de les modifier.
6. Ne pas utiliser `T` et `F` à la place de `TRUE` et `FALSE`.