Exercices - 3

Préparation

1. Vérifiez que les options globales de RStudio sont conformes aux options recommandées dans le cours.

2. Créez un projet RStudio intitulé `cours_r_partie_3`. Ce projet sera créé vide. Nous initialiserons un projet git en cochant la case adéquate.

Exercice

Pour cet exercice, nous allons utiliser le même jeu de données issu de https://data.nantesmetropole.fr/. Si besoin, le jeu de données est re-téléchargeable ici. L'objectif de cet exercice est d'apprendre à écrire des fonctions. Nous utiliserons régulièrement `git` pour apprendre à faire des `commits`/`push` et prendre de bons automatismes.

1. Créez un dossier `data/`, et copiez le jeu de données à l'intérieur du dossier. Initialisez un script `.R` ou Quarto à la racine du projet. Créez un premier commit comprenant l'intégralité des modifications, **en excluant le fichier csv**. Initiez un nouveau projet sur GitHub vide (sans `README`) que vous appellerez `cours_r_partie_3`. Dans un terminal, dans le projet RStudio, configurez le serveur distant en utilisant l'expression suivante :

```
git remote add origin <l'adresse de votre répertoire>
```

Faites un `push`. Commentez. Dans la suite, nous ferons un commit intitulé *question X* après la résolution de chaque question, suivi d'un `push` sur le serveur distant.

Nous allons désormais implémenter plusieurs fonctions de statistiques descriptives relatives aux trajets vélos.

2. Importez le jeu de données téléchargé, et nommez l'objet `df_velo`. Créez deux `data.frame` contenant les informations relatives à vos deux boucles favorites, nommés sur le format `df_b_numero` (*e.g* `df_b_788` pour la boucle de 50 Otages Nord), ainsi que deux `data.frames` comprenant les informations relatives à vos deux jours favoris, nommés selon le format `df_d_date` (*e.g* `df_d_0105` pour la boucle du 1er mai). Nous utiliserons ces `data.frames` pour faire des tests dans la suite.

3. Implémentez une fonction `filtre_anomalie()`. Cette fonction prendra en entrée un `data.frame` respectant le même schéma que notre jeu de données `df_velo`. Elle retournera le même `data.frame` sans les lignes contenant des anomalies `Forte` ou `Faible`. Appliquez cette fonction sur les `data.frame` définis ci-dessus. Combien de lignes sont filtrées ?

4. Implémentez une fonction `compter_nombre_trajets()`. Cette fonction prendra en entrée un `data.frame` respectant le même schéma que notre jeu de données `df_velo`. Elle comptera

le nombre total de trajets référencés. Appliquez cette fonction sur les `data.frame` définis ci-dessus.

5. Implémentez une fonction `compter_nombre_boucle()`. Cette fonction comptera le nombre de boucles dans un `data.frame`. Appliquez également sur les quatre `data.frame`. Pouviez-vous prédire certains résultats ?

6. Implémentez une fonction `trouver_trajet_max()`. Cette fonction identifiera quel est la paire boucle-jour avec le maximum de passages décomptés. Elle retournera un objet contenant :

- Le *nom* de la boucle
- Le *jour* correspondant au maximum de passage enregistrés
- Le *nombre de passages* enregistrés

Appliquez aux quatre `data.frame`.

7. Appliquer la fonction `trouver_trajet_max()` au jeu de donnée complet `df_velo`. Quelle est votre analyse ? Proposez une modification de la fonction `filtre_anomalie()`.

8. Implémentez une fonction `calcul_distribution_semaine()`. Cette fonction calculera la somme des trajets référencés pour chaque jour de la semaine. Appliquez aux quatre `data.frame`. Pouvez-vous prédire certains résultats ?

9. Implémentez une fonction `plot_distribution_semaine()`. Cette fonction prendra en entrée un `data.frame` respectant le même schéma que notre jeu de données `df_velo`. Elle comptera, pour chaque jour de la semaine, le nombre de trajets détectés. Elle utilisera pour cela les fonctions `filtre_anomalie()` et `calcul_distribution_semaine()`. Nous utiliserons un *diagramme en colonne* pour représenter ces informations. Appliquez aux quatre `data.frame`.

10. Améliorer la fonction `trouver_trajet_max()` pour qu'elle filtre les anomalies et qu'elle retourne, en plus des éléments précédents :

- La *moyenne par jour* des passages sur cette boucle
- La *moyenne par boucle* des passages pour cette date

Appliquez aux quatre `data.frame` et au jeu de donnée complet `df_velo`.

11. *(Bonus)* Appliquer la classe `trajet` au jeu de donnée et créer sa fonction générique `max.trajet()`. Cette fonction prendra en entrée un paramètre `x` qui sera notre objet de type `trajet`. Elle vérifiera que le `data.frame` en entrée contient les colonnes `Boucle de comptage`, `Jour` et `Total`. Elle retournera l'ensemble des informations données par `trouver_trajet_max()`.

Pour tester cette fonction, vous affecterez la classe `trajet` au l'un des quatre `data.frame`.

Aide pour l'exercice

- Utilisez la fonction d'aide `?` pour déterminer les bons paramètres de lecture.
- La lecture d'un fichier `csv` peut se faire en `base R` avec `read.delim` ou avec `readr` et la fonction `read_delim`.
- La sous-sélection de données peut se faire via la fonction `filter` de `dplyr`.

- Les fonctions `group_by` et `summarise` de `dplyr` peuvent vous aider à calculer des sommes et des moyennes.
- La carte des boucles et disponible ici.