Programmation de spécialité (python)
TD 1-2 : pandas et matplotlib

Julien Velcin

2024-2025

Partie 1 : Chargement des données

1.1 Commencez par charger en mémoire un jeu de données issues du site des données ouvertes par le Grand Lyon sur les randonnées recensées dans la région. Pour cela, rendez-vous sur le site et téléchargez le fichier de description des données au format json :

https://data.grandlyon.com/jeux-de-donnees/boucles-randonnee-metropole-lyon/donnees

Le fichier s'appelle `evg_esp_veg.envpdiprboucle.json` (attention à ne pas prendre le geo-json !). Pour charger le jeu de données avec Python, utilisez la fonction `load` de la librairie `json` sur le fichier que vous devez ouvrir en lecture avec la fonction `open`. Enregistrez le résultat dans la variable **d_json**. N'hésitez pas à vous aider du site suivant :

https://www.programiz.com/python-programming/json

1.2 La variable que vous venez de charger est un dictionnaire, vous pouvez le vérifier avec la commande `type`. Observez les clefs (commande `keys`) et essayez d'avoir accès à la liste des champs/variables du jeu de données et à la liste des randonnées (càd le tableau de données proprement dit).

1.3 Enregistrez la liste des champs dans une variable **var** et les randonnées dans une variable **rando**. Vérifiez le nombre d'enregistrements et essayez d'avoir accès aux valeurs.

1.4 Quelle est la taille du jeu de données, càd le nombre de lignes ? Quel est le nom de la randonnée de la ligne 10 ?

Partie 2 : Quelques traitements avec pandas

2.1 Pour rendre ces données pleinement utilisables, nous allons convertir le fichier `json` en format tabulaire avec la librairie `pandas`. Il vous suffit de créer un `DataFrame` à partir de la variable **rando** à l'aide de la fonction `from_dict`. N'hésitez pas à vérifier les dimensions de votre tableau à l'aide de l'attribut `shape` de l'object que vous venez de créer.

2.2 `df.head()` et `df.tail()` permettent d'avoir un aperçu (resp.) du début et la fin de l'objet `DataFrame`. Essayez ces commandes.

2.3 Les commandes `loc` et `iloc` permettent d'accéder facilement à une partie du tableau, par exemple :

`df.iloc[:, 1:4]` signifie qu'on veut accéder à toutes les lignes (:) et aux colonnes 1 à 4 (1:4).

Avec `loc`, il est possible d'accéder aux colonnes via leur nom. C'est surtout intéressant pour faire du changement de valeur dans une colonne à l'aide d'une condition, par ex. `df.loc[df.type`

`== "test", "type"] = 0`. Au contraire, `iloc` fonctionne uniquement avec des index entiers. Testez cette fonctionnalité.

2.4 Accéder aux données d'une colonne devient aisée puisqu'il suffit de mentionner le nom de la variable, avant d'y appliquer différents traitements (ex. calculer une moyenne, un max, etc.) :
`df["nom de la colonne"]`, auquel on applique les commandes `mean()`, `max()`...

Pour tester cela, utilisez la commande `value_counts()` qui calcule la distribution des valeurs pour une variable catégorielle. Vous testerez avec la variable `difficulte` de la randonnée.

2.5 Pour manipuler certaines données, il peut être nécessaire de faire des traitements préalables. Par exemple, il est nécessaire pour la variable `temps_parcours` de retirer la fin de la chaîne de caractère (en l'occurrence les 4 derniers caractères " min"). Commencez par visualiser la colonne correspondante.

2.6 Construisez une nouvelle série en transformant ces valeurs en nombres entiers correspondants, débarassés de l'unité de mesure. Une solution élégante consiste à utilisation une liste "compréhensible" (*list comprehension*) mais vous pouvez construire une solution plus simple avec une boucle.

2.7 A présent que la variable a bien été convertie en numérique, vous pouvez appliquer les opérateurs classiques comme la moyenne à la série (fonction `mean()`).

La librairie pandas permet des traitements plus avancés, inspirés des bases de données. Il est ainsi possible de regrouper les lignes en fonction des valeurs d'une colonne (opération **group by**) puis réaliser des opérations statistiques.

2.8 Calculez le temps de parcours moyen pour les randonnées de chaque niveau de difficulté.

Pour vous aider, n'hésitez pas à aller voir sur le site :
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html

Partie 3 : Diagrammes en barre et camemberts avec matplotlib

3.1 Pour commencer, on peut vouloir afficher un diagramme en barre sur le contenu d'une série, par exemple la distribution selon une certaine variable. Pour cela, il suffit d'invoquer la fonction `plot.bar()` à partir d'une série pandas. Affichez le diagramme correspondant au nombre de randonnées pour chaque niveau de difficulté (facile, moyen difficile).

3.2 On peut vouloir ordonner en fonction de l'ordre alphabétique des index (ici, les noms des catégories). Dans ce cas, il faut utiliser la fonction `sort_index()` sur la série comportant les données à afficher. Affichez à nouveau le diagramme en barres mais avec un tri sur les index.

3.3 Affichez le diagramme sous forme de camembert à l'aide de la commande `plot.pie()`.

Partie 4 : Diagramme de dispersion avec matplotlib

4.1 Pour commencer, nettoyez la colonne correspondant à la variable longueur afin qu'elle contiennent une variable réelle. L'opération est très similaire à celle sur le temps, mais il faudra aussi remplacer la chaîne d'un nombre en français (ex. 12,5) par l'équivalent en anglais (ex. 12.5). Pour cela, vous pouvez utiliser la fonction `replace` des chaînes de caractères.

4.2 A présent, vous pouvez croiser les deux variables avec la commande `scatter` :

```
df.plot.scatter(x = nom\_var_1, y = nom_var_2)
```

4.3 La commande `scatter` retourne un objet qui permet de modifier les caractéristiques du graphique, comme son titre, ses axes, etc.

```
ax = df.plot.scatter(x = nom_var_1, y = nom_var_2)
ax.set_title("...")
ax.set_xlabel(...")
(etc.)
```

Améliorez la lisibilité de votre figure en lui donnant un titre et en précisant les unités des deux axes.

4.4 Pour finir, calculer le coefficient de corrélation R entre les deux séries, par ex. en passant par la commande `corr` appliquée aux séries pandas (càd aux colonnes).

Il est bien sûr possible d'afficher plusieurs séries, avec chacune son label et sa couleur, voire d'attribuer des formes ou des gradients de couleur aux points. Pandas reposant sur Matplotlib, les possibilités sont nombreuses.

Pour plus de visualisations avec Pandas :
https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html
La page suivante donne une galerie assez exhaustive des possibilités :
https://matplotlib.org/gallery/index.html
Une autre bibliothèque très efficace pour la visualisation s'appelle `seaborn` (cf. https://seaborn.pydata.org/).

Partie 5 : Analyse sur plusieurs tables

Pour finir, on souhaite pouvoir réaliser des analyses sur plusieurs tables liées entre elles, comme on le ferait avec une base de données relationnelle.

Nous allons travailler sur les accidents ayant eu lieu en 2018 et accessibles en format CSV via la plateforme nationale data.gouv.fr :
https://www.data.gouv.fr/fr/datasets/base-de-donnees-accidents-corporels-de-la-circulation/

Il s'agit en particulier des fichiers `lieux-2018.csv`, `vehicules-2018.csv`, `usagers-2018.csv`, `caracteristiques-2018.csv`.

On vous conseille fortement de jeter un œil au fichier qui décrit ces données, fourni au format PDF (`description-des-bases-de-donnees-onisr-annees-2005-a-2018.pdf`).

5.1 Commencez par charger les données en mémoire à l'aide de la commande `read_csv` de la bibliothèque `pandas`. Si vous rencontrez des problèmes d'encodage, une solution simple consiste à sauvegarder les fichiers en UTF-8 à l'aide d'un éditeur de texte avant de passer sur Python.

5.2 Calculez le nombre d'accidents ayant eu lieu à Lyon (code département 690) sur la période. Pour cela, passez par la création d'une table DataFrame **accidents_lyon**.

5.3 Calculez le nombre d'accidents ayant eu lieu impliquant une bicyclette (code véhicule 01) sur la période. Pour cela, passez par la création d'une table DataFrame **accidents_velo**.

5.4 Réalisez une jointure à l'aide de l'opérateur `join` entre les deux tables afin de calculer le nombre d'accidents impliquant un vélo ayant eu lieu à Lyon pendant la période.

5.5 Construisez une visualisation sous forme d'histogramme permettant de comparer le nombre d'accidents ayant eu lieu à vélo dans plusieurs grandes villes de France (à vous de choisir lesquelles).

5.6 S'il vous reste du temps, n'hésitez pas à pousser plus loin vos investigations. Par exemple, écartez le biais sur la taille de la ville avec une normalisation adaptée puis refaites vos analyses en travaillant de manière relative. Un autre type d'analyse consiste à charger plusieurs fichiers en mémoire correspondant à des années différentes afin d'étudier l'évolution du nombre d'accidents dans le temps.