TP 8 – Analyse exploratoire de données avec Python

S.Gibet

Année 2019-2020

Objectifs du TP : comprendre le contenu et la structure des données, les problèmes (données manquantes, aberrantes).
Les exercices suivants permettent de vous initier à la manipulation des données à travers la librairie *panda*. Plus précisément, ils permettent d'explorer ces données, de les transformer et de les visualiser. La plupart des exemples en Python sont fournis, il s'agit de bien les comprendre, voire de tester par vous même d'autres opérations d'analyse et de manipulation. Certaines questions sont à chercher en regardant dans la documentation relative à panda.

Récupération des données

Données

MovieLens 1M Data Set contient les notes attribuées à des films par des utilisateurs du site Movielens. Les données sont fournies, mais peuvent être trouvées, si besoin, à l'adresse: `http://grouplens.org/datasets/movielens/`

Packages

```
import pandas as pd   #pour l'exploration de données
import numpy as np    #pour les opérations numériques
```

Questions

Lecture des données

Avant d'aborder les questions qui suivent, il faut lire lire le *readme* afin de connaitre les différents fichiers de données et leur format : *users.dat*, *ratings.dat* et *movies.dat*.

1. Lire les données "users" dans un DataFrame Pandas et afficher les 5 premières valeurs. Vous pourrez utiliser le code Python suivant :

   ```
   unames = ['user_id', 'gender', 'age', 'occupation', 'zip']
   users = pd.read_csv('ml-1m/users.dat', sep='::', header=None,
   names=unames, engine='python')
   users.head()
   ```

2. Lire les données "ratings" dans un DataFrame Pandas et afficher les 10 premières valeurs.

3. Lire les données "movies" dans un DataFrame Pandas et afficher les 10 premières valeurs.

4. Fusionner les données des 3 fichiers dans un seul DataFrame.

```
data = pd.merge((pd.merge(users, ratings), movies)
data.head()
```

Exploration des données

Dans les exercices suivants vous suivrez attentivement le tutoriel qui explique les fonctionnalités de l'opération *groupby*, qui permet de *splitter* des objets, de les combiner ou d'appliquer une fonction. `https://www.tutorialspoint.com/python_pandas/python_pandas_groupby.htm`

1. Combien de films ont une note supérieure à 4.5 ? Existe-t-il une différence entre les hommes et les femmes?

```
np.sum(data.rating > 4.5)
np.sum(data.rating[data.gender == 'F'])> 4.5)
np.sum(data.rating[data.gender == 'M'] > 4.5)
```

2. Même question en regardant les proportions : par exemple le nombre de films notés plus de 4.5 par des femmes sur le nombre total de films notés par des femmes (idem pour des hommes).

3. Combien de films ont une note médiane au-dessus de 4.5 parmi les hommes de plus de 30 ans ? et parmi les femmes de plus de 30 ans ?

```
np.sum(data[(data.gender == 'M') & (data.age >= 30)].
groupby('movie_id', axis=0)['rating'].median() >= 4.5)
```

4. Quels sont les films les plus populaires? Afficher les caractéristiques du film le plus populaire.

```
data.groupby('movie_id', axis=0)['rating'].mean().nlargest(15)
data[data.movie_id == 787]
```

5. Quels sont les films qui sont suffisamment populaires ? Vous construirez un sous-ensemble de données en fonction du "rating", et afficherez les premières valeurs dans un tableau contenant les moyennes des ratings et le nombre de ratings.

```
data.groupby('movie_id', axis=0)['rating'].count().head()
data2 = pd.concat([data.groupby('movie_id', axis=0)['rating'].mean(),
                   data.groupby('movie_id', axis=0)['rating'].count()], axis=1)
data2.columns = ['mean_rating', 'n_rating']
data2.head()
```

6. On va définir une popularité minimale (en fonction du nombre de notes obtenues), et garder uniquement les films qui sont au-dessus d'un seuil. Vous afficherez ensuite le titre des 2 films les plus populaires dans ce nouvel ensemble. Vous pourrez utiliser la fonction *sort* ainsi que le paramètre *ascending*.

7. Quel est le film qui est le plus souvent noté par les utilisateurs ? Pour cela vous trierez le DataFrame selon n_rating.

Visualisation des données

1. Afficher l'histogramme des notes de tous les films.

   ```
   data.rating.hist(bins=5, align='left', range=[1, 6])
   ```

2. Afficher l'histogramme du nombre de notes reçues par chaque film

   ```
   data.groupby('movie_id', axis=0)['rating'].count().hist(bins=10)
   ```

3. Afficher l'histogramme des notes moyennes des films. La distribution des notes dépend-elle du sexe?

4. Afficher l'histogramme des notes des films qui ont été notés plus de 30 fois (regarder la documentation de map, dict)

   ```
   map_id_to_count = data.groupby('movie_id')['rating'].count().to_dict()
   data['movie_count'] = data['movie_id'].map(map_id_to_count)
   # rajoute une variable comptant le nombre de votes par film

   # changer kde par hist pour retrouver un histogramme et non une estimation de la densité
   data[data.movie_count >= seuil_pop].groupby('movie_id', axis=0)['rating'].
   mean().plot(kind='kde', color='b')
   data[data.movie_count <= seuil_pop].groupby('movie_id', axis=0)['rating'].
   mean().plot(kind='kde', color='g')
   ```

5. Afficher un "scatter plot" des notes moyennes pour les hommes contre les notes des femmes pour chaque film (notés plus de 100 fois). Vous regarderez la documentation des opérations *pivot_ table*, et l'affichage en mode "scatter" (plot(..., kind='scatter', ...))

6. Afficher un "scatter plot" des notes moyennes des hommes vs les femmes pour chaque film noté moins de 100 fois.

7. Quelle disparité de comportement observe-t-on entre les hommes et les femmes d'après la dernière figure ?

Pour lire des données dans des répertoires gérés par vous – librairie python glob

Le module *glob* trouve tous les *pathnames* qui correspondent à un patron spécifié suivant les règles utilisées par le shell Unix. Les caractères utilisés sont : *, ?, et les caractères dont l'étendue est exprimée par []. Voir la documentation glob :
`https://docs.python.org/2/library/glob.html`
Cette librairie pourra être utilisée dans le prochain TP.