Aller plus loin avec la librairie Pandas

Table des matières

I. La gestion des fichiers

Objectifs de ce cours

- Comprendre la particularité de la gestion de fichiers avec Pandas

Il est assez commun de travailler sur des fichiers de données, notamment avec des fichiers `csv`, `excel` ou bien `json`.

Lire et écrire dans les fichiers est une tâche qui peut être complexe, pour des raisons d'encodage notamment. Pandas va donc nous simplifier la vie en nous mettant à disposition des fonctions qui vont s'occuper de lire le fichier suivant son extension et de le transformer en objet `DataFrame` afin de pouvoir le traiter avec les fonctions vu précédemment.

Pour les `csv` on utilisera la fonction `read_csv()`, pour les `json` la fonction `read_json()` et pour les fichiers `excel` la fonction `read_excel()`. Dans la plupart des cas, les fonctions de type `read()` prennent un chemin (`path`) relatif ou absolu vers les données en question.

```
pokemon = pd.read_csv('../data/pokemon.csv'); pokemon.head(3)
```

La méthode `read_json` fonctionne sur le même principe.

```
books = pd.read_json('../data/books.json'); books.head(3)
```

Si la fonction `read_excel` ne fonctionne pas c'est sans doute qu'il vous faut un paquet en plus `xlrd`. Pour l'installer il vous faut lancer la commande suivante dans *une cellule de code*.

```
!pip install xlrd
```

Attention aux arguments des fonctions de type `read()`, il est parfois nécessaire de leur préciser un certain nombre d'arguments relatifs à l'encodage (les header, index, feuilles excel à importer, caractères séparateurs, etc.)

```
xl = pd.read_excel('../data/SampleData.xlsx',sheet_name='SalesOrders'); xl.head(5)
```

Résultat obtenu (Out) :

	OrderDate	Region	Rep	Item	Units	Unit Cost	Total
0	2018-01-06	East	Jones	Pencil	95	1.99	189.05
1	2018-01-23	Central	Kivell	Binder	50	19.99	999.50
2	2018-02-09	Central	Jardine	Pencil	36	4.99	179.64
3	2018-02-26	Central	Gill	Pen	27	19.99	539.73
4	2018-03-15	West	Sorvino	Pencil	56	2.99	167.44

On peut vérifier que ces 3 objets sont bien du type `DataFrame` avec la fonction native `type` de Python.

```
type(pokemon)==type(xl)==type(books)==pd.core.frame.DataFrame
```

Out : `True`

II. La gestion des valeurs manquantes

Objectifs de ce cours

- Comprendre ce que sont les valeurs manquantes
- Savoir gérer les valeurs manquantes

La gestion des valeurs manquantes est une des principale problématique dans le domaine de la science de données. Dans le domaine statistique une valeur manquante représente une absence d'observation. En informatique elles sont représentées par des `NaN`.

Comme vous pouvez vous en douter les méthodes adaptées pour la gestion de ces valeurs font l'objet de multiples recherches. On appelle **imputation** les méthodes utilisées pour remplacer ces données manquantes.

Les méthodes d'imputation les plus simples consistent à remplacer les données manquantes par leur moyenne ou leur médiane.

A la recherche des NaN

En effet, la première tâche à effectuer est de rechercher les `NaN`.

On peut utiliser pour cela les fonctions `isna()` et `isnull()`. Dans la suite on va former un sous `DataFrame` avec le `DataFrame` pokemon afin d'illustrer nos exemples.

```
pokemon.isnull()
```

On voit que les fonctions `isna()` et `isnull()` nous renvoient des `DataFrame`, ce qui est moyennement pratique pour une lecture claire. Heureusement nous savons maintenant bien manipuler ces objets, il est donc temps de mettre notre savoir en pratique.

```
pokemon.isnull().sum();
```

```
pokemon.isna().sum().sort_values(ascending=False)[0:4]
```

Résultat obtenu (Out) :

```
type2 384
percentage_male 98
height_m 20
weight_kg 20
dtype: int64
```

```
poke_df_null = pokemon[pokemon.isna().sum().sort_values(ascending=False)[0:5].index.tolist()]
poke_df_null.tail(100)
```

Out :

	type2	percentage_male	height_m	weight_kg	is_legendary
701	fairy	50.0	0.2	2.2	0
702	fairy	NaN	0.3	5.7	0
703	NaN	50.0	0.3	2.8	0
704	NaN	50.0	0.8	17.5	0
705	NaN	50.0	2.0	150.5	0
...	...	...	...	...	...
796	fairy	NaN	9.2	999.9	1
797	steel	NaN	0.3	0.1	1
798	dragon	NaN	5.5	888.0	1
799	NaN	NaN	2.4	230.0	1
800	fairy	NaN	1.0	80.5	1

100 rows x 5 columns

Maintenant que nous avons bien identifié les `NaN` dans notre `DataFrame` nous allons voir comment les gérer.

Les méthodes dropna() et fillna()

Ces méthodes ont des noms assez explicites, elles sont à manipuler avec précaution.

Elles ont plusieurs options, notamment sur la façon d'enlever les `NaN`. On peut voir notamment l'argument `how` de la fonction `dropna()` qui par défaut est `how=all` ce qui signifie que vous allez supprimer uniquement les lignes avec que des `NaN`. Si vous voulez l'inverse, c'est-à-dire supprimer les lignes qui contiennent au moins un `NaN` il vous faut passer comme paramètre `any`.

```
1 poke_df_null.dropna(how='any')
```

	type2	percentage_male	height_m	weight_kg	is_legendary
0	poison	88.1	0.7	6.9	0
1	poison	88.1	1.0	13.0	0
2	poison	88.1	2.0	100.0	0
5	flying	88.1	1.7	90.5	0
11	flying	50.0	1.1	32.0	0
...	...	...	...	...	...
777	fairy	50.0	0.2	0.7	0
778	psychic	50.0	0.9	19.0	0
779	dragon	50.0	3.0	185.0	0
782	fighting	50.0	1.2	47.0	0
783	fighting	50.0	1.6	78.2	0

339 rows × 5 columns

On peut aussi supprimer des `NaN` en fonction d'une colonne particulière avec l'argument `subset=['nom de la colonne']`.

```
1 poke_df_null.dropna(how='any', subset=['height_m'])
```

	type2	percentage_male	height_m	weight_kg	is_legendary
0	poison	88.1	0.7	6.9	0
1	poison	88.1	1.0	13.0	0
2	poison	88.1	2.0	100.0	0
3	flying	88.1	1.7	90.5	0
4	flying	50.0	1.1	32.0	0
...	...	...	...	...	...
796	fairy	50.0	0.2	0.7	0

778	psychic	50.0	0.9	19.0	0
779	dragon	50.0	3.0	185.0	0
782	fighting	50.0	1.2	47.0	0
783	fighting	50.0	1.6	78.2	0

781 rows × 5 columns

Voyons maintenant comment remplir ces valeurs à l'aide de la méthode `fillna()`.

```
poke_mean = poke_df_null.weight_kg.mean()
```

```
poke_df_null.weight_kg.fillna('mean').isnull().sum() ==
poke_df_null.weight_kg.fillna(poke_mean).isna().sum()
```

Out : `True`

Dans le cas des variables catégorielles on a juste besoin de spécifier la valeur à remplacer.

```
poke_df_null.type2.fillna('Unknown')
```

Out :

```
0 poison
1 poison
2 poison
3 Unknown
4 Unknown
...
796 flying
797 steel
798 dragon
799 Unknown
800 fairy
Name: type2, Length: 801, dtype: object
```

III. La gestion des dates

Objectifs de ce cours

- Comprendre la problématique de gestion des dates en machine
- Savoir convertir des données en type date

La gestion des dates est un problème classique en informatique, il faut donc être précautionneux quant à l'utilisation de celles-ci.

Un exemple marquant est le bug de l'an 2038[1].

On va encore une fois se reposer sur la librairie Pandas et sa fonction `to_datetime()` qui va nous servir à caster les variables de la `Series` comme des `datetime64`.

1 https://fr.wikipedia.org/wiki/Bug_de_l%27an_2038

```
df = pd.DataFrame({'year': [2015, 2016],
                   'month': [2, 3],
                   'day': [4, 5]})
```

```
df.dtypes
```

Résultat obtenu (Out) :

```
year int64
month int64
day int64
dtype: object
```

```
pd.to_datetime(df)
```

Out :

```
0 2015-02-04
1 2016-03-05
dtype: datetime64[ns]
```

Exemple sur les données extraites de la feuille excel.

```
xl.OrderDate = pd.to_datetime(xl.OrderDate); #xl.OrderDate
```

Un argument important dans cette méthode c'est `format` suivant le format de vos données.

```
pd.to_datetime(df,format='%Y%m%d')
```

Out :

```
0 2015-02-04
1 2016-03-05
dtype: datetime64[ns]
```

IV. La visualisation avec Matplotlib

Objectifs de ce cours

- Savoir visualiser graphiquement un jeu de données Pandas

Il peut être utile de visualiser de façon graphique les colonnes/valeurs d'un tableau. Nous allons utiliser pour cela la librairie `matplotlib` qui est très *« pandas friendly »*, il est donc nécessaire d'importer la librairie afin d'accéder aux commandes `pandas` qui l'encapsulent.

On utilisera la fonction `figure(figsize=(12,6))` qui va nous permettre de grandir la figure.

```
plt.figure(figsize=(12,6))
```

Nous utiliserons dans la suite des exemples le `dataset` importé depuis excel nommé `xl`.

```
xl.head()
```

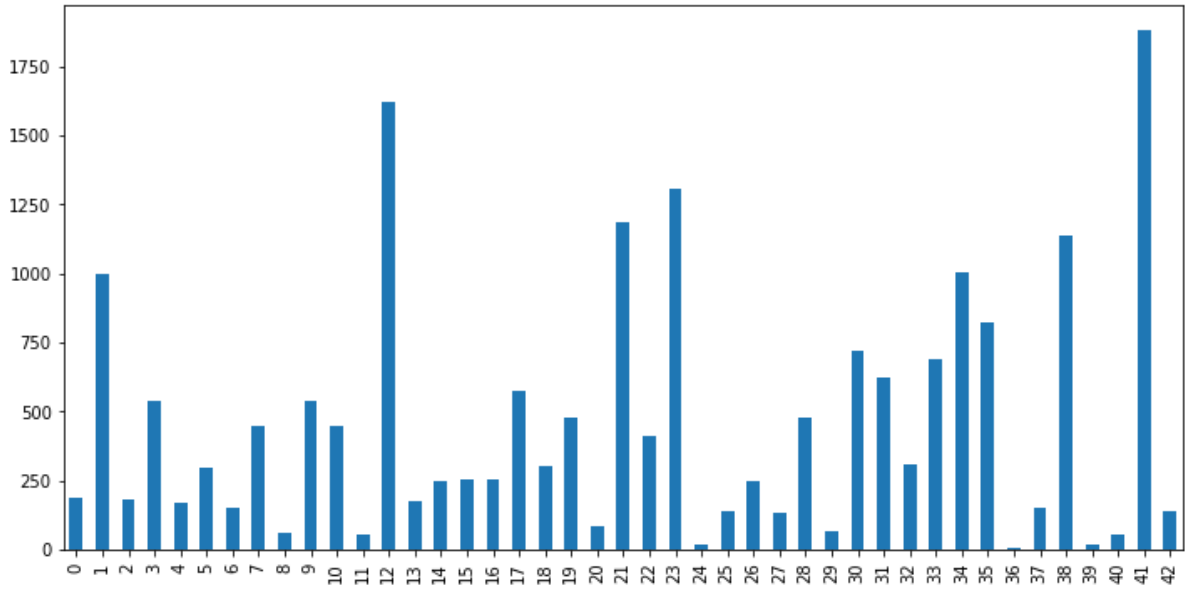
Résultat obtenu (Out) :

	OrderDate	Region	Rep	Item	Units	Unit Cost	Total
0	2018-01-06	East	Jones	Pencil	95	1.99	189.05
1	2018-01-23	Central	Kivell	Binder	50	19.99	999.50
2	2018-02-09	Central	Jardine	Pencil	36	4.99	179.64
3	2018-02-26	Central	Gill	Pen	27	19.99	539.73
4	2018-03-15	West	Sorvino	Pencil	56	2.99	167.44

Globalement, la fonction `plot` prend en argument `kind` le type de graphique.

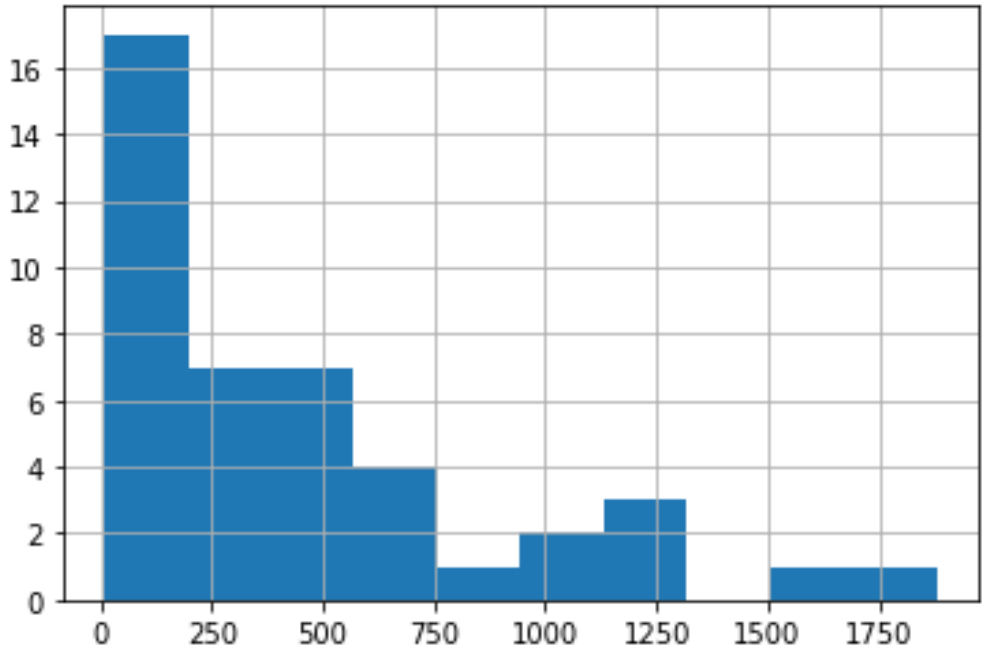
Ici on va regarder la distribution de la colonne `Total` du `DataFrame`, ce qui correspond au nombre de ventes totales à une certaine date.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
xl.Total.plot(kind='bar');
```



On peut aussi tracer un histogramme de cette variable avec la fonction `hist()` afin d'observer sa distribution.

```
xl.Total.hist();
```



On peut voir que malgré la praticité on peut être assez vite limité quant à la gestion de nos graphiques, notamment sur le "style".

Complément **Exercice**

Vous trouverez l'exercice sous format Jupyter en fichier téléchargeable ci-dessous.

[cf. Pandas.ipynb]

Exercice

Exercice

Si `df` est un `DataFrame`, que fait la fonction `df.dtypes` ?

- ○ Elle renvoie le type du DataFrame
- ○ Elle modifie le type du DataFrame
- ○ Elle donne le type de la première colonne
- ○ Elle renvoie le type de données que contient chaque colonne

Exercice

Identifiez l'énoncé correct.

- ○ Pandas remplace les données manquantes par des `NaN` par défaut
- ○ Les `series` sont similaires à des tableaux `Numpy`
- ○ Les deux réponses ci-dessus
- ○ Aucune de ces réponses

Exercice

Que fait l'argument `nrows` de la méthode `pandas.read_csv()` ?

- ☐ Il limite l'import des colonnes
- ☐ Il limite l'import des lignes
- ☐ Il charge les `nrows` lignes du fichier
- ☐ Il donne le nombre minimum de dimensions du DataFrame

Exercice

Quel est le délimiteur pour les fichiers `.csv` et `.tsv` ?

- ○ Le caractère virgule (,)
- ○ Tout caractère tel que la virgule (,) ou la tabulation (\t) utilisé pour séparer les colonnes
- ○ Tout caractère tel que la virgule (,) ou la tabulation (\t) utilisé pour séparer les lignes
- ○ Aucun

Exercice

Parmi les options suivantes, laquelle peut être utilisée pour créer un `DataFrame` ?

- ○ Des scalaires
- ○ Un `ndarray`
- ○ Un dictionnaire
- ○ Les trois