Index

Semaine 4

Ali Assi, PhD

Nous avons déjà vu qu'une série a un index (pour ses éléments) et qu'un dataframe en a deux (un pour les lignes et un second pour les colonnes). Nous avons vu comment `.loc`, ainsi que les sélecteurs de lignes et les sélecteurs de colonnes, peuvent être très puissants.

Mais les index dans Pandas sont bien plus flexibles que ce que nous avons vu jusqu'à présent :

1. Il est possible de transformer une colonne existante en index, ou de retransformer un index en une colonne normale.
2. Nous pouvons combiner plusieurs colonnes en un "multi-index" hiérarchique, puis effectuer des recherches sur des parties spécifiques de cette hiérarchie. En effet, savoir comment créer, interroger et manipuler des cadres de données multi-indexés est la clé d'un travail fluide avec Pandas.
3. Nous pouvons également créer des "tableaux croisés dynamiques", dans lesquels les lignes et les colonnes ne reflètent pas nos données d'origine, mais plutôt des résumés agrégés de ces données.

Dans ce chapitre, nous allons nous entraîner à utiliser toutes ces techniques, afin de mieux comprendre comment créer, modifier et manipuler une variété de types d'index. Après ces exercices, vous saurez mieux comment utiliser les index Pandas pour récupérer des données de manière plus souple et plus facile.

Quelques fonctions utiles

1. **pd.set_index**, renvoie un nouveau dataframe avec un nouvel index. Par exemple: `df = df.set_index('name')`

In [1]:
```
import pandas as pd

student_dict = {'Name': ['Pierre', 'Ribal', 'Stephane'], 'Age': [22, 21, 18],

# create DataFrame from dict
student_df = pd.DataFrame(student_dict)
student_df
```

Out[1]:

	Name	Age	Weight
0	Pierre	22	85
1	Ribal	21	77
2	Stephane	18	54

In [2]:
```
# set index using column
student_df = student_df.set_index('Name')
student_df
```

Out[2]:

	Age	Weight
Name		
Pierre	22	85
Ribal	21	77
Stephane	18	54

2. **pd.reset_index**, renvoie un nouveau dataframe avec un index par défaut (numérique, positionnel). Par exemple, `df = df.reset_index()`

In [3]:
```
import pandas as pd
import numpy as np

student_dict = {'Name': ['Pierre', 'Ribal', np.NaN, 'Stephane'], 'Age': [22, 2

# create DataFrame from dict
student_df = pd.DataFrame(student_dict)
student_df
```

Out[3]:

	Name	Age	Weight
0	Pierre	22.0	85.0
1	Ribal	21.0	77.0
2	NaN	NaN	NaN
3	Stephane	18.0	54.0

In [4]:
```
# drop NA
student_df = student_df.dropna()
student_df
```

Out[4]:

	Name	Age	Weight
0	Pierre	22.0	85.0
1	Ribal	21.0	77.0
3	Stephane	18.0	54.0

In [5]:
```
# reset index
student_df = student_df.reset_index()
student_df
```

Out[5]:

	index	Name	Age	Weight
0	0	Pierre	22.0	85.0
1	1	Ribal	21.0	77.0
2	3	Stephane	18.0	54.0

In [6]:
```
## Comment utiliser le MultiIndex dans Pandas - Exemple de Gapminder dataset
```

In [5]:
```
!pip install Gapminder
```

```
Collecting Gapminder
  Using cached gapminder-0.1-py3-none-any.whl (32 kB)
Requirement already satisfied: pandas in c:\users\admin\miniconda3\envs\proje
t1\lib\site-packages (from Gapminder) (1.1.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\admin\minic
onda3\envs\projet1\lib\site-packages (from pandas->Gapminder) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in c:\users\admin\miniconda3\envs
\projet1\lib\site-packages (from pandas->Gapminder) (2021.3)
Requirement already satisfied: numpy>=1.15.4 in c:\users\admin\miniconda3\env
s\projet1\lib\site-packages (from pandas->Gapminder) (1.20.3)
Requirement already satisfied: six>=1.5 in c:\users\admin\miniconda3\envs\pro
jet1\lib\site-packages (from python-dateutil>=2.7.3->pandas->Gapminder) (1.1
5.0)
Installing collected packages: Gapminder
Successfully installed Gapminder-0.1

WARNING: Ignoring invalid distribution -rotobuf (c:\users\admin\miniconda3\en
vs\projet1\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\admin\miniconda3\en
vs\projet1\lib\site-packages)
```

In [6]:
```
import pandas as pd
from gapminder import gapminder
gapminder.head()
```

Out[6]:

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

Nous allons fixer l'index sur le 'continent', 'country', 'year'. Et on va utiliser `inplace equals (=) true` pour sauvegarder ces modifications.

In [8]:
```
gapminder.set_index(['continent', 'country', 'year'], inplace = True)
```

Maintenant, nous avons l'index ici, ('continent', 'country', 'year') et c'est notre multi-index.

In [9]:
```
gapminder.head()
```

Out[9]:

			lifeExp	pop	gdpPercap
continent	**country**	**year**			
		1952	28.801	8425333	779.445314
		1957	30.332	9240934	820.853030
Asia	**Afghanistan**	**1962**	31.997	10267083	853.100710
		1967	34.020	11537966	836.197138
		1972	36.088	13079460	739.981106

In [15]:
```
# Supposons, par exemple, que nous voulions tout ce qui se trouve dans le Europ
gapminder.loc['Europe']
```

Out[15]:

		lifeExp	pop	gdpPercap
country	year			
	1952	55.230	1282697	1601.056136
	1957	59.280	1476505	1942.284244
Albania	**1962**	64.820	1728137	2312.888958
	1967	66.220	1984060	2760.196931
	1972	67.690	2263554	3313.422188
...	**...**	...	...	...
	1987	75.007	56981620	21664.787670
	1992	76.420	57866349	22705.092540
United Kingdom	**1997**	77.218	58808266	26074.531360
	2002	78.471	59912431	29478.999190
	2007	79.425	60776238	33203.261280

360 rows × 3 columns

In [19]:
```
# Supposons que nous ne voulions que les données du Royaume-Uni en 1997.
gapminder.loc[('Europe', 'United Kingdom', 2007)]
```

Out[19]:
```
lifeExp      7.942500e+01
pop          6.077624e+07
gdpPercap    3.320326e+04
Name: (Europe, United Kingdom, 2007), dtype: float64
```

On peut utiliser gapminder.unstack pour remodeler cet ensemble de données.

In [20]:
```
gapminder_pivot = gapminder.unstack()
gapminder_pivot
```

Out[20]:

		life									
	year	1952	1957	1962	1967	1972	1977	1982	1987	1992	1
continent	country										
Africa	Algeria	43.077	45.685	48.303	51.407	54.518	58.014	61.368	65.799	67.744	69
	Angola	30.015	31.999	34.000	35.985	37.928	39.483	39.942	39.906	40.647	40
	Benin	38.223	40.358	42.618	44.885	47.014	49.190	50.904	52.337	53.919	54
	Botswana	47.622	49.618	51.520	53.298	56.024	59.319	61.484	63.622	62.745	52
	Burkina Faso	31.975	34.906	37.814	40.697	43.591	46.137	48.122	49.557	50.260	50
...	...	...	...	...	...	...	...	...	...	...	...
Europe	Switzerland	69.620	70.560	71.320	72.770	73.780	75.390	76.210	77.410	78.030	79
	Turkey	43.585	48.079	52.098	54.336	57.005	59.507	61.036	63.108	66.146	68
	United Kingdom	69.180	70.420	70.760	71.360	72.010	72.760	74.040	75.007	76.420	77
Oceania	Australia	69.120	70.330	70.930	71.100	71.930	73.490	74.740	76.320	77.560	78
	New Zealand	69.390	70.260	71.240	71.520	71.890	72.220	73.840	74.320	76.330	77

142 rows × 36 columns

Imaginons maintenant que nous voulions le faire dans la direction opposée. Tout ce que j'aurais à faire, c'est de dépivoter et nous allons faire `gapminder_pivot` .

In [21]:
```
gapminder_pivot = gapminder_pivot.stack()
gapminder_pivot
```

Out[21]:

			lifeExp	pop	gdpPercap
continent	**country**	**year**			
		1952	43.077	9279525	2449.008185
		1957	45.685	10270856	3013.976023
Africa	**Algeria**	**1962**	48.303	11000948	2550.816880
		1967	51.407	12760499	3246.991771
		1972	54.518	14760787	4182.663766
...	**...**	**...**	...	...	...
		1987	74.320	3317166	19007.191290
		1992	76.330	3437674	18363.324940
Oceania	**New Zealand**	**1997**	77.550	3676187	21050.413770
		2002	79.110	3908037	23189.801350
		2007	80.204	4115771	25185.009110

1704 rows × 3 columns

3. **s.isin**, renvoie une série booléenne indiquant si une valeur dans `s` est un élément de l'argument. Par exemple, `s.isin(['A', 'B', 'C')`

In [15]:
```
import pandas as pd

#creating pandas Series
series = pd.Series([5, 8, 8, 4, 7, 3, 2, 1, 4, 6])

series
```

Out[15]:
```
0    5
1    8
2    8
3    4
4    7
5    3
6    2
7    1
8    4
9    6
dtype: int64
```

In [16]:
```
# Apply isin() function to check for the specified values
result = series.isin([3, 6])
result
```

Out[16]:
```
0    False
1    False
2    False
3    False
4    False
5     True
6    False
7    False
8    False
9     True
dtype: bool
```

Exercice

Nous avons déjà vu de nombreux exemples de la façon de récupérer une ou plusieurs lignes d'un dataframe en utilisant son index, ainsi que l'attribut `loc`. Nous n'avons pas nécessairement besoin d'utiliser l'index pour sélectionner des lignes dans un datafarme, mais cela facilite la compréhension et rend le code plus clair. C'est pourquoi nous voulons souvent utiliser l'une des colonnes existantes d'un dataframe comme index. Parfois, nous voulons le faire de façon permanente, alors que d'autres fois, nous voulons le faire brièvement, juste pour rendre nos requêtes plus claires.

Dans cet exercice, je vais vous demander d'effectuer quelques requêtes sur un autre ensemble de données de la ville de New York, un ensemble qui a suivi toutes les contraventions de stationnement au cours de l'année 2020 - plus de 12 millions d'entre elles. En théorie, vous pouvez effectuer ces requêtes sans modifier l'index du dataframe. Cependant, je souhaite que vous vous entraîniez à définir et à réinitialiser l'index. Nous allons le faire souvent dans ce chapitre, et vous le ferez probablement aussi souvent lorsque vous travaillerez avec Pandas sur des ensembles de données réels.

C'est dans cet esprit que je vous invite à:

1. Créez un dataframe à partir du fichier `nyc-parking-violations-2020.csv`. Seule une partie des colonnes nous intéresse :
   A. $Date\ First\ Observed$
   B. $Plate\ ID$
   C. $Registration\ State$
   D. $Issue\ Date$ (une chaîne au format $MM/JJ/AAAA$, toujours suivie de $12:00:00$ AM)
   E. $Vehicle\ Make$
   F. $Street\ Name$
   G. $Vehicle\ Color$

In [17]:
```
import pandas as pd
from pandas import Series, DataFrame
```

In [19]:
```
filename = './data/nyc-parking-violations-2020.csv'

#filename2 = 'C:/Users/Admin/Desktop/programming2/semaine4/data/nyc-parking-vi

df = pd.read_csv(filename, usecols = ['Date First Observed', 'Plate ID', 'Regi
                                      'Vehicle Make', 'Street Name', 'Vehicle
df.columns
```

Out[19]:
```
Index(['Plate ID', 'Registration State', 'Issue Date', 'Vehicle Make',
       'Street Name', 'Date First Observed', 'Vehicle Color'],
      dtype='object')
```

2. Définissez l'index du dataframe comme étant la colonne $Issue\ Date$.

In [ ]:
```

```

3. Quelles sont les trois marques de véhicules les plus susceptibles d'être sanctionnées par une contravention le 2 janvier ?. Aide: utiliser la fonction value_count() et head()

In [ ]:
```

```

4. Dans quelles cinq rues les voitures ont-elles reçu le plus de contraventions le 1er juin 2020 ?Aide: utiliser la fonction value_count() et head()

In [ ]:
```

```

5. Définissez maintenant l'index comme étant la colonne $Vehicle\ Color$.

In [ ]:
```

```

6. Quelle était la marque la plus courante des voitures sanctionnées qui étaient soit bleues, soit rouges ? Aide: utiliser la fonction value_count() et head()

In [ ]:
```

```

Comme nous l'avons vu, la définition de l'index peut faciliter la création de requêtes sur nos données. Mais il arrive que nos données soient de nature hiérarchique. C'est là que le concept Pandas de `"multi-index"` entre en jeu. Avec un `multi-index`, vous pouvez définir l'index non pas sur une seule colonne, mais sur plusieurs colonnes. Imaginez, par exemple, un dataframe contenant des données sur les ventes : Vous pourriez souhaiter que les ventes soient décomposées par année, puis par région. Dès que vous utilisez l'expression " décomposées par ", il est presque certain qu'un multi-index est une bonne idée. (Voir l'encadré ci-dessus, "Travailler avec des multi-index", pour une description plus complète).

Exercice 2

Dans cet exercice, nous allons examiner un résumé des résultats du SAT, un test standardisé d'admission à l'université largement utilisé aux États-Unis. Le fichier CSV (`sat-scores.csv`) comporte 99 colonnes et 577 lignes, décrivant les 50 États américains et trois États non américains (Porto Rico, les Îles Vierges et Washington, DC), de 2005 à 2015.

Dans cet exercice, je vous demande de:

1. Lire le fichier de scores en ne conservant que les colonnes Année, Code de l'Etat, Total Mathématiques, Total Participants et Total Verbal.

In [ ]:

2. Créer un multi-index basé sur l'année et le code à deux lettres de l'État.

In [ ]:

3. Combien de personnes ont passé le SAT en 2005 ?

In [ ]:

4. Quel était le score moyen en mathématiques au SAT en 2010 dans les états de New York, du New Jersey, du Massachusetts et de l'Illinois ?

In [ ]:

5. Quel était le score moyen au SAT oral en 2012-2015 en Arizona, en Californie et au Texas ?

In [ ]: