seance1

December 12, 2021

1 TD 1

Nous travaillerons sur un jeu de données issues de la base d'exemples de MATLAB qui se trouvent dans le fichier Carsmall3.xlsx. Ces mesures concernent des paramètres de voitures des années 1970 à 1982.

1.1 Présentation du jeu de données

[ ]:
```
import pandas as pd

df = pd.read_excel("Carsmall3.xlsx", index_col=0)
df
```

[ ]:

Model	Model_Year	Origin	Weight	MPG	\
chevrolet chevelle malibu	70	USA	3504	18.0	
buick skylark 320	70	USA	3693	15.0	
plymouth satellite	70	USA	3436	18.0	
amc rebel sst	70	USA	3433	16.0	
ford torino	70	USA	3449	17.0	
…	…	…	…	…	
ford mustang gl	82	USA	2790	27.0	
volkswagen pickup	82	Europe	2130	44.0	
dodge rampage	82	USA	2295	32.0	
ford ranger	82	USA	2625	28.0	
chevrolet s-10	82	USA	2720	31.0	

Model	Horsepower	Displacement	Acceleration
chevrolet chevelle malibu	130	307.0	12.0
buick skylark 320	165	350.0	11.5
plymouth satellite	150	318.0	11.0
amc rebel sst	150	304.0	12.0
ford torino	140	302.0	10.5
…	…	…	…
ford mustang gl	86	140.0	15.6
volkswagen pickup	52	97.0	24.6
dodge rampage	84	135.0	11.6

```
ford ranger                                79          120.0          18.6
chevrolet s-10                             82          119.0          19.4

[280 rows x 7 columns]
```

Le tableau contient des données de types hétérogènes: des chaines de caractère (type `object`), des entiers (`int64`) et des flottants (`float64`). Cependant la colonne Model_Year tient plus d'une donnée qualitative que d'une donnée quantitative, malgré son type scalaire.

[ ]:
```
print(df.dtypes)
```

```
Model_Year        int64
Origin           object
Weight            int64
MPG             float64
Horsepower        int64
Displacement    float64
Acceleration    float64
dtype: object
```

Pour vérifier qu'il n'y a pas de données manquantes, on peut utiliser la fonction `isnull()`, qui nous renvoie un DataFrame contenant pour chaque cellule un booléen indiquant si la cellule est vide ou invalide. Ensuite, on peut aggréger les informations et obtenir si *une au moins une* cellule est vide ou invalide avec `values.any()`

[ ]:
```
print(f"Le tableau comporte des données manquantes: {df.isnull().values.any()}")
```

```
Le tableau comporte des données manquantes: False
```

1.2 Description des données

On peut obtenir un aperçu global rapide avec la fonction `describe`:

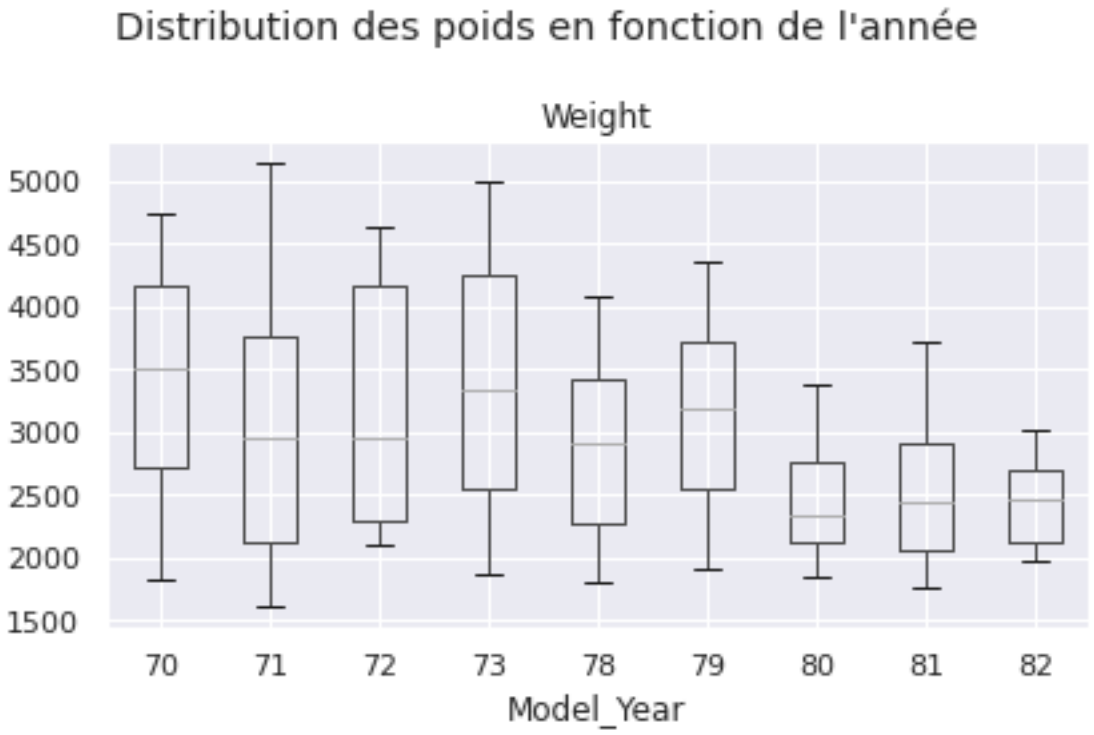
[ ]:
```
df.describe()
```

[ ]:
```
       Model_Year       Weight         MPG  Horsepower  Displacement  \
count  280.000000   280.000000  280.000000  280.000000    280.000000
mean    76.035714  2968.257143   23.888571  107.189286    197.862500
std      4.382943   846.448133    8.404399   41.421730    109.302431
min     70.000000  1613.000000    9.000000   46.000000     68.000000
25%     72.000000  2233.000000   17.000000   76.000000    105.000000
50%     78.000000  2789.500000   23.000000   92.000000    151.000000
75%     80.000000  3542.000000   30.000000  137.250000    302.500000
max     82.000000  5140.000000   46.600000  230.000000    455.000000

       Acceleration
count    280.000000
mean      15.305357
std        2.951873
```

```
min        8.000000
25%       13.475000
50%       15.150000
75%       17.125000
max       24.800000
```

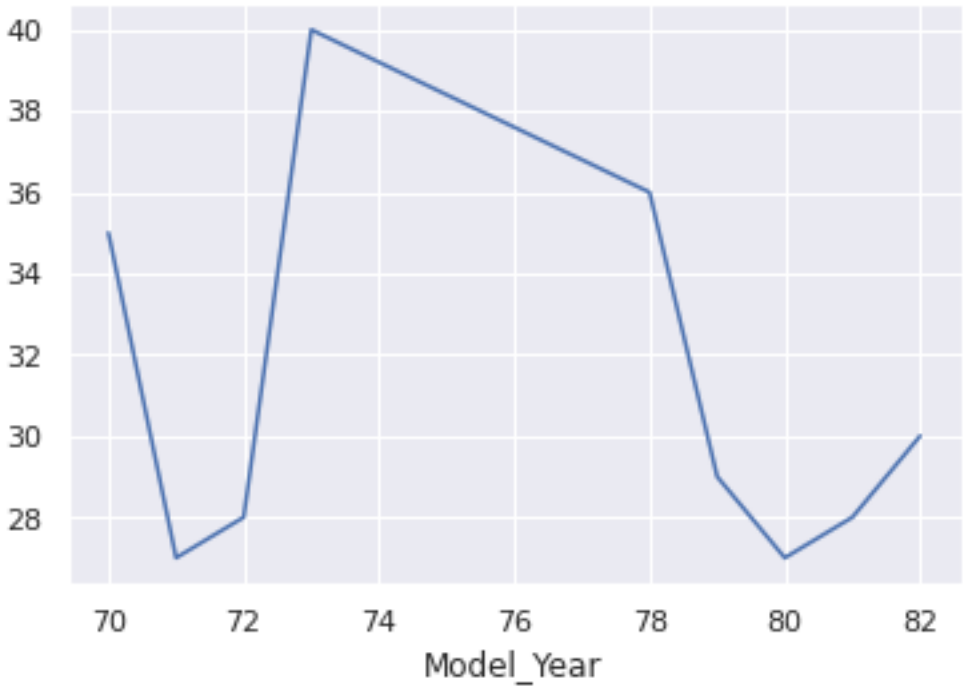
[ ]:
```
import seaborn as sns
import pylab as pl
sns.set()

ax = df.boxplot(column="Weight", by="Model_Year")
pl.tight_layout()
pl.suptitle("Distribution des poids en fonction de l'année")
ax
```

[ ]:
```
<AxesSubplot:title={'center':'Weight'}, xlabel='Model_Year'>
```



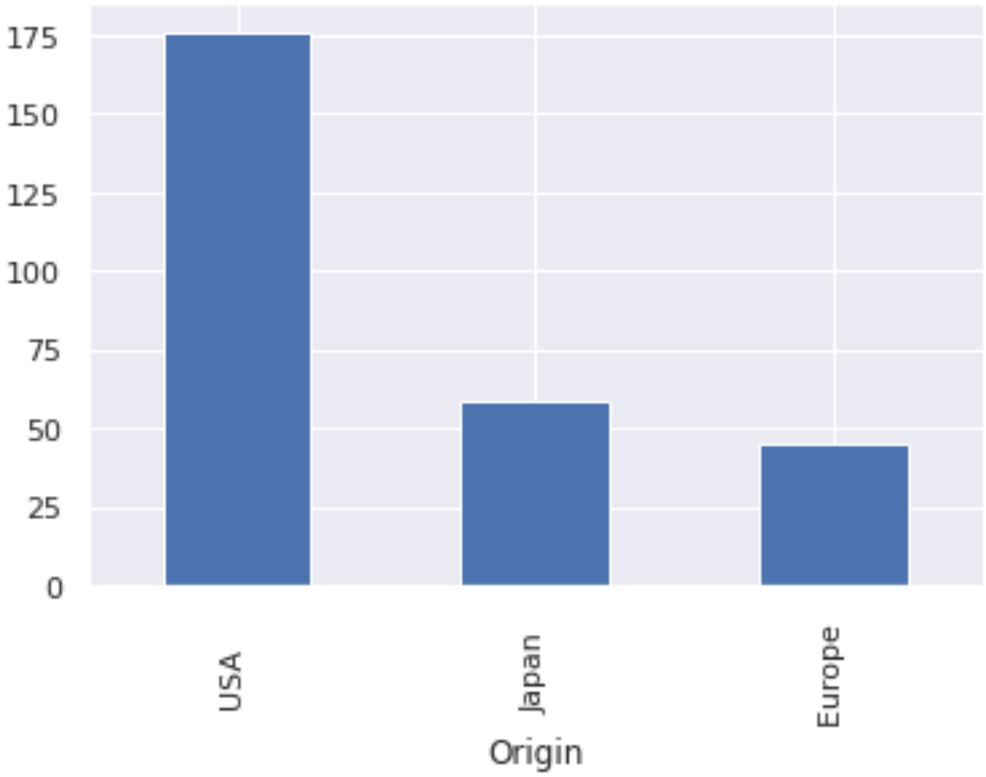
[ ]:
```
df.value_counts("Model_Year").sort_index().plot.line()
```

[ ]:
```
<AxesSubplot:xlabel='Model_Year'>
```



[ ]:
```
df.value_counts("Origin").plot(kind="bar")
```

[ ]:
```
<AxesSubplot:xlabel='Origin'>
```



[ ]:
```
pd.crosstab(df.Model_Year, df.Origin)
```

[ ]:
```
Origin      Europe   Japan    USA
Model_Year
70               6       2       27
71               4       4       19
72               5       5       18
73               7       4       29
78               6       8       22
79               4       2       23
80               8      13        6
81               3      12       13
82               2       9       19
```