

Introduction à l'exploration de données

Statistiques descriptives



Analyse exploratoire

Objectif : Obtenir une vision global d'un jeu de données en recherchant des régularités, des relation entre variables, ou même des groupes homogènes. 3 approches :

- Univariée : statistique descriptive pour chaque variables
- Bivariée : étude des relations entre variables (stat croisées, khi-2, indices de corrélation)
- Multidimensionnel : permet d'observer la présence possible de groupes d'individus homogènes (ACP, AFC, ACM)

Analyse exploratoire

L'analyse exploratoire à aussi pour but de **détecter les valeurs rare ou manquante**.

Celle-ci peuvent ce voir accorder une importance plus grande qu'elle ne devrait en avoir.

De ce fait elle déséquilibre l'analyse et la modélisation qui en découle si on cherche à en faire une.

Dans le cas des valeurs manquante il est préférable de laisser l'observation de coté.

Le contexte de l'analyse

Nous étudions une **population**, c'est-à-dire un ensemble dont les éléments sont des **individus** au sens statistique : humains, animaux, plantes, objets ou... concepts.

Ces individus sont définis par des variables de différents types.

Analyse exploratoire

Les types de données

Type de données	Equivalent/Exemple
Quantitatives continues	float/La distance domicile Simplon
Quantitatives discrètes	Integer/L'age
Qualitatives nominales	String/La couleur des yeux
Qualitatives ordinales	Integer/Classement de la ligue des champion

Analyse univarié

- Moyenne
- Médiane
Valeur de la série qui permet de couper l'ensemble des valeurs en deux parties égales : mettant d'un côté une moitié des valeurs, et de l'autre côté l'autre moitié des valeurs.
- Quartiles
La population peut être partagée en quatre sous-populations de même effectif. Les valeurs qui correspondent aux limites du partage sont des quartiles.
- Mode
Le mode d'une série qualitative ou discrète est la modalité ou la valeur qui enregistre le plus grand effectif.

Analyse univarié

- la variance
mesure de la dispersion des valeurs d'un échantillon autour de sa moyenne. Elle exprime la moyenne des carrés des écarts à la moyenne
- L'écart-type
Racine carré de la variance

Les graphiques

Un graphique est une représentation de données statistiques. Et s'il existe plusieurs types de données, il y a aussi différents types de graphiques.

Sont les **données** qui déterminent le type de graphique qui les représente et non le chargé d'études

Les graphiques

- Le diagramme circulaire est à privilégier pour représenter des séries dont le caractère est qualitatif. Les parts du diagramme ont des aires qui sont proportionnelles aux effectifs de chaque modalité.
- Le diagramme en barres est parfait lorsque le caractère est quantitatif discret. Il est également utilisé pour les caractères qualitatifs. C'est la longueur de chaque barre qui est proportionnelle aux effectifs ou aux pourcentages.

Les graphiques

- La courbe est très souvent utilisée pour représenter une série quantitative en situation d'évolution, par exemple dans le temps. En réalité ce n'est pas une courbe puisqu'elle relie de façon rectiligne les points correspondant aux observations.
- Le nuage de points est idéal pour représenter des individus en fonction de deux critères quantitatifs, matérialisés par deux axes.

Mise en pratique

```
# Import de la librairie pandas avec l'allias pd
```

```
import pandas as pd
```

```
#Création d'un dataframe
```

```
df = pd.DataFrame({'Colonne1': [1.1, 2.7, 5.3], 'Colonne2': [2, 10, 9], 'Colonne3': ['oui', 'non', 'non']},  
                  index = ['a', 'b', 'c'])
```

```
print(df)
```

	Colonne1	Colonne2	Colonne3
a	1.1	2	oui
b	2.7	10	non
c	5.3	9	non

Mise en pratique

```
# Import de données depuis un fichier Excel dans un dataframe

nom_colonnes = ['consommation', 'cylindres', 'deplacement',
                'puissance', 'poids', 'acceleration', 'annee', 'origine', 'nom']

data = pd.read_excel('Desktop\Auto-mpg.xlsx',
                    index_col=None, header=None, names=nom_colonnes)
```

```
# Afficher les premières lignes du dataframe
print(data.head())
```

	consommation	cylindres	deplacement	puissance	poids	acceleration	\
0	18.0	8	307.0	130.0	3504	12.0	
1	15.0	8	350.0	165.0	3693	11.5	
2	18.0	8	318.0	150.0	3436	11.0	
3	16.0	8	304.0	150.0	3433	12.0	
4	17.0	8	302.0	140.0	3449	10.5	

	annee	origine	nom
0	70	1	chevrolet chevelle malibu
1	70	1	buick skylark 320
2	70	1	plymouth satellite
3	70	1	amc rebel sst
4	70	1	ford torino

Mise en pratique

```
# Afficher les dimesions du dataframe  
print(data.shape)  
print(data.shape[0])  
print(data.shape[1])
```

```
(398, 9)  
398  
9
```

```
#informations sur les données  
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 398 entries, 0 to 397  
Data columns (total 9 columns):  
consommation      398 non-null float64  
cylindres          398 non-null int64  
deplacement        398 non-null float64  
puissance          392 non-null float64  
poids              398 non-null int64  
acceleration       398 non-null float64  
annee              398 non-null int64  
origine            398 non-null int64  
nom                398 non-null object  
dtypes: float64(4), int64(4), object(1)  
memory usage: 28.1+ KB  
None
```

Mise en pratique

```
#description des données  
print(data.describe())
```

	consommation	cylindres	deplacement	puissance	poids \
count	398.000000	398.000000	398.000000	392.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623
std	7.815984	1.701004	104.269838	38.491160	846.841774
min	9.000000	3.000000	68.000000	46.000000	1613.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000

	acceleration	annee	origine
count	398.000000	398.000000	398.000000
mean	15.568090	76.010050	1.572864
std	2.757689	3.697627	0.802055
min	8.000000	70.000000	1.000000
25%	13.825000	73.000000	1.000000
50%	15.500000	76.000000	1.000000
75%	17.175000	79.000000	2.000000
max	24.800000	82.000000	3.000000

Mise en pratique

```
# Fréquences corisés
```

```
print(pd.crosstab(data['cylindres'],data['annee']))
```

annee	70	71	72	73	74	75	76	77	78	79	80	81	82
cylindres													
3	0	0	1	1	0	0	0	1	0	0	1	0	0
4	7	13	14	11	15	12	15	14	17	12	25	21	28
5	0	0	0	0	0	0	0	0	1	1	1	0	0
6	4	8	0	8	7	12	10	5	12	6	2	7	3
8	18	7	13	20	5	6	9	8	6	10	0	1	0

```
# Pourcentage corisés
```

```
print(pd.crosstab(data['cylindres'],data['annee'], normalize='index'))
```

annee	70	71	72	73	74	75	\
cylindres							
3	0.000000	0.000000	0.250000	0.250000	0.000000	0.000000	
4	0.034314	0.063725	0.068627	0.053922	0.073529	0.058824	
5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
6	0.047619	0.095238	0.000000	0.095238	0.083333	0.142857	
8	0.174757	0.067961	0.126214	0.194175	0.048544	0.058252	

Mise en pratique

```
#accès à une ou plusieurs colonne  
print(data[['nom', 'consommation']].head())
```

	nom	consommation
0	chevrolet chevelle malibu	18.0
1	buick skylark 320	15.0
2	plymouth satellite	18.0
3	amc rebel sst	16.0
4	ford torino	17.0

```
# Affiche les valeurs 10 à 12  
print(data['nom'][10:13])
```

```
10    dodge challenger se  
11    plymouth 'cuda 340  
12    chevrolet monte carlo  
Name: nom, dtype: object
```

```
: # Accées à la valeur de la première ligne, première colonne  
print(data.iloc[0, 0])  
# Accées à la valeur de la dernière ligne, première colonne  
print(data.iloc[-1, 0])
```

```
18.0  
31.0
```


Mise en pratique

```
# Toutes les colonnes avec les ligne de 0 à 4
print(data.iloc[0:5,:])
```

	consommation	cylindres	deplacement	puissance	poids	acceleration	\
0	18.0	8	307.0	130.0	3504	12.0	
1	15.0	8	350.0	165.0	3693	11.5	
2	18.0	8	318.0	150.0	3436	11.0	
3	16.0	8	304.0	150.0	3433	12.0	
4	17.0	8	302.0	140.0	3449	10.5	

	annee	origine	nom
0	70	1	chevrolet chevelle malibu
1	70	1	buick skylark 320
2	70	1	plymouth satellite
3	70	1	amc rebel sst
4	70	1	ford torino

```
# Liste des voitures dont son de l'année 80
print(data.loc[data['annee']==80,:].head())
```

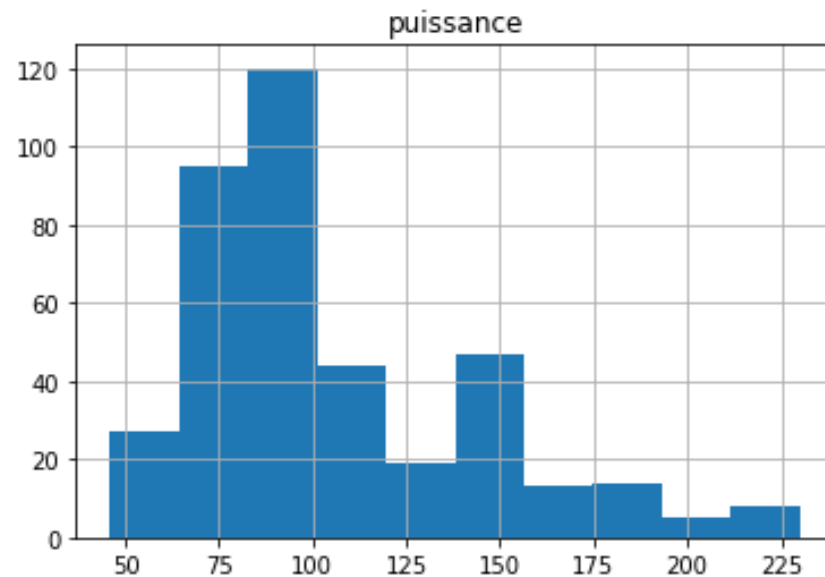
	consommation	cylindres	deplacement	puissance	poids	acceleration	\
309	41.5	4	98.0	76.0	2144	14.7	
310	38.1	4	89.0	60.0	1968	18.8	
311	32.1	4	98.0	70.0	2120	15.5	
312	37.2	4	86.0	65.0	2019	16.4	
313	28.0	4	151.0	90.0	2678	16.5	

Mise en pratique

```
# Pour que les graphique apparassent dans le notebook
%matplotlib inline
#importation de la librairie
import matplotlib.pyplot as plt
```

```
# Histogramme de la puissance
data.hist(column='puissance')
```

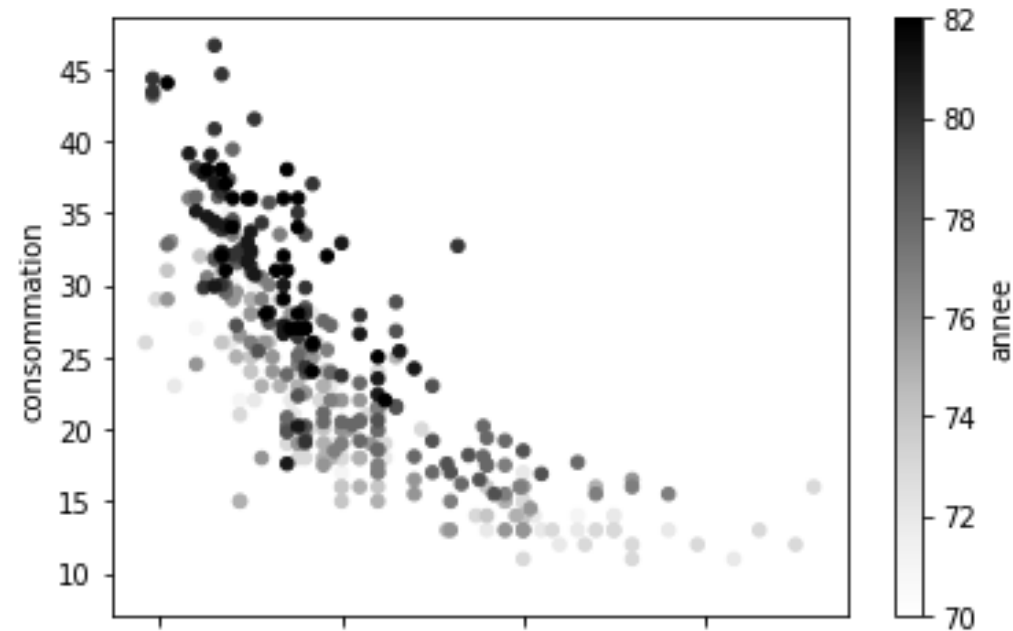
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000248D498DF60>]],
      dtype=object)
```



Mise en pratique

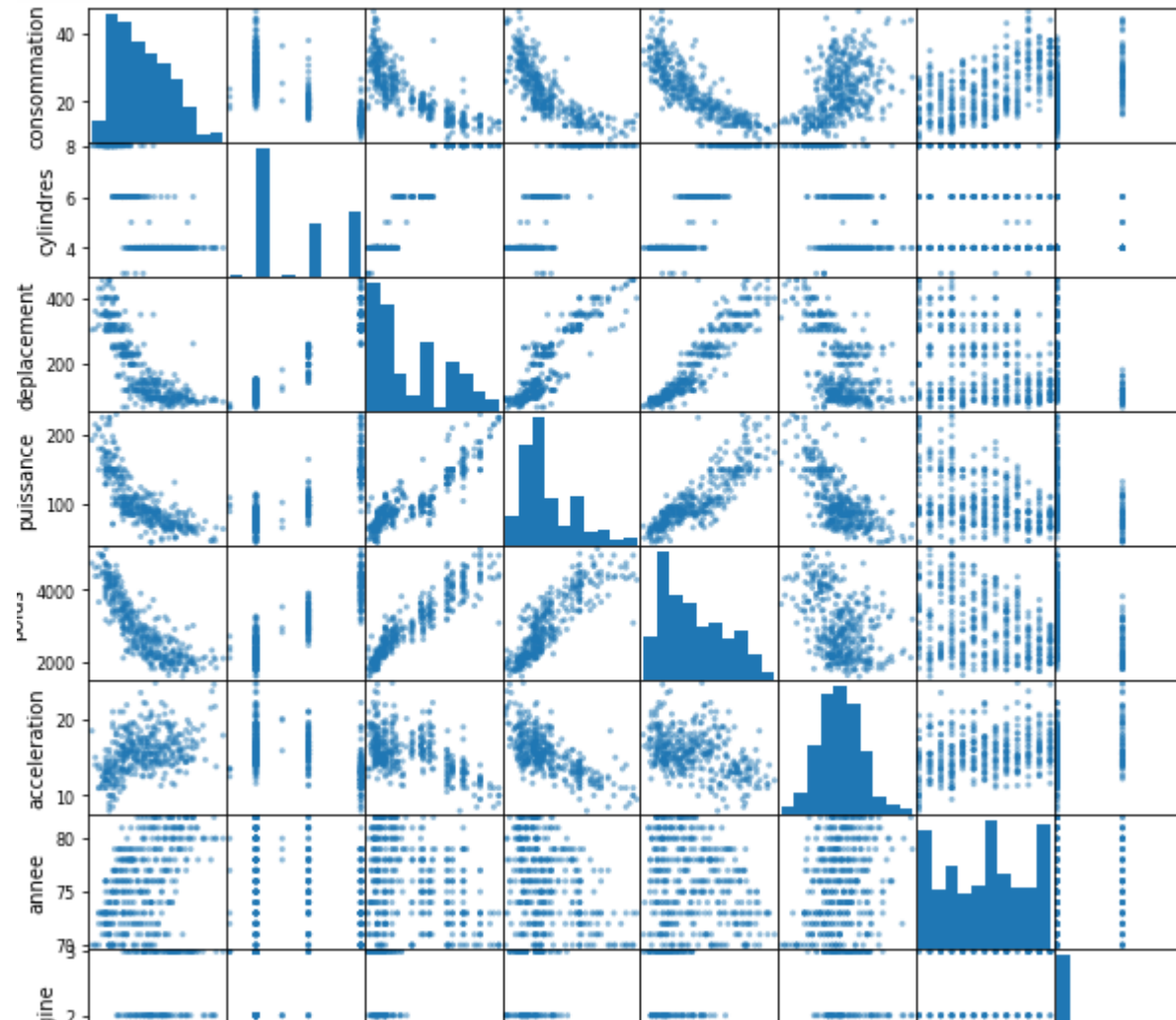
```
# Nuage de point : consommation en fonction de la puissance  
# avec niveau de gris selon l'année  
data.plot.scatter(x='puissance',y='consommation',c='annee')
```

<matplotlib.axes._subplots.AxesSubplot at 0x248d8654630>



Mise en pratique

```
pd.plotting.scatter_matrix(data.select_dtypes(exclude=['object']), figsize=(10, 10))
```



Pratique

- A vous d'explorer Pandas et Matplotlib.
 - Importer les données avec Pandas
 - Produire les statistiques décrivant les données
 - Produire les graphique adapter au données