TP 13 – Statistiques

I Avec la bibliothèque pandas

Le fichier titanic.csv est un tableur contenant les données des passagers du Titanic. Nous allons l'étudier avec Python en utilisant la bibliothèque pandas (voir documentation jointe).

```
import pandas as pd
titanic = pd.read_csv('titanic.csv')
```

```
titanic
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

- PassengerId (identifiant passager)
- Survived (0 : décédé, 1 : a survécu)
- Pclass (classe, de 1 à 3)
- Name (Nom, prénom et titre)
- Sex (male/female)
- Age (en années)
- SibSp (nombre de frère, soeur, beau-frère, belle soeur, mari ou femme)
- Parch (nombre de parents et d'enfants)
- Ticket (numéro du billet)
- Fare (prix du billet)
- Cabin (numéro de cabine)
- Embarked (port d'emarquement : C - Cherbourg, S - Southampton, Q = Queenstown)

Exercice 1 En utilisant la documentation pandas (donnée en annexe) :

1. Afficher les colonnes et leur type (`int` (entier), `float` (nombre décimal), `object` (donnée non numérique)).

```
titanic.info()
```

2. Afficher (un aperçu de) la liste des passagers en donnant uniquement leur nom (il s'agit ici de récupérer et d'afficher les données uniquement de la colonne correspondante).

```
titanic['Name']
```

3. Calculer la proportion des passagers qui a survécu au naufrage. On pourra :
   - Récupérer le nombre de passagers survivants en faisant la somme des éléments de la colonne 'Survived' (il s'agit d'abord de récupérer les données de la colonne 'Survived' puis d'utiliser une commande permettant de faire la somme de toutes ses données)
   - Récupérer le nombre de passagers au total en comptant le nombre d'éléments de la colonne 'Names'
   - Puis, on effectue le rapport, qu'on multiplie ensuite par 100 pour obtenir le pourcentage.

```
nbresurvivants = titanic['Survived'].sum()
nbrepassager = titanic['Name'].count()
proportionsurvivant = (nbresurvivants/nbrepassager)*100
```

```
Out [0]: 38,38383838383838
```

4. Calculer cette même proportion mais uniquement pour les passagers de première classe. On commence par filter les données pour n'avoir que celles correspondant aux passagers de première classe.

```
#On commence par filtrer pour ne récupérer
#que les données de la première classe
titanic1 = titanic[ titanic['Pclass']== 1 ]
#Puis on calcule le taux de survivant comme à
la question précédente en utilisant ces nvelles données
(titanic1['Survived'].sum()/titanic1['Name'].count())*100
```

```
Out [0]: 62,96296296296297
```

5. Comparer maintenant le taux de survie chez les hommes et les femmes.

```
hommes = titanic[ titanic['Sex'] == 'male' ]
proph = (hommes['Survived'].sum()/hommes['Name'].count())*100
print('Pour les hommes', proph)

femmes = titanic[ titanic['Sex'] == 'female' ]
propf = (femmes['Survived'].sum()/femmes['Name'].count())*100
print('Pour les femmes', propf)
```

```
Out [0]: Pour les hommes 18,890814558058924
         Pour les femmes 74,20382165605095
```

6. Tracer l'histogramme de répartition des âges dans le navire.

```
titanic.hist(column = 'Age')
```

7. Déterminer le prix moyen du billet par classe de voyage.

```
for classe in [1,2,3]:
    #filtre
    titanic_classe = titanic[ titanic['Pclass'] == classe ]
    moyenne = titanic_classe['Fare'].mean()
    print('Prix moyen en classe', classe, ':', moyenne)
```

Out [0]:
```
Prix moyen en classe 1 : 84.1546875
Prix moyen en classe 2 : 20.662183152173913
Prix moyen en classe 3 : 13.675550101832993
```

II Statistiques "à la main"

On va analyser à la main les résultats (sur 20) d'un DS d'une classe de 30 élèves. Les notes sont données dans la liste suivante.

Entrée [1]:
```
notes = [11, 16, 18, 15, 16, 17, 13, 14, 13, 12, 10, 15, 20, 13, 14,
15, 15, 11, 17, 12, 15, 13, 19, 16, 12, 16, 14, 15, 14, 14]
```

① Ecrire une fonction `moyenne` prenant en argument une liste `L` et qui renvoie la **moyenne** de cette liste. *On rappelle que la moyenne des valeurs $(x_k)_{k=1,\dots,n}$ est donnée par*

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

Entrée [2]:
```
#Version 1
def moyenne(L):
    moy = 0
    for k in range(len(L)):
        moy = moy + L[k]
    moy = moy/len(L)
    return(moy)

#Version 2
def moyenne(L):
    moy = 0
    for x in L:
        moy = moy + x
    moy = moy/len(L)
    return(moy)
```

Entrée [3]:
```
moyenne(notes)
```

Out [3]:
```
14.5
```

② Ecrire une fonction `ecarttype` prenant en argument une liste `L` et qui renvoie l'**écart-type** de cette liste. *On rappelle que l'écart-type des valeurs $(x_k)_{k=1,\dots,n}$ est donnée par*

$$s_x = \frac{1}{n}\sum_{k=1}^{n} x_k^2 - (\overline{x})^2$$

où $\overline{x}$ désigne la moyenne de ces données.

Entrée [4]:
```
def ecarttype(L):
    e = 0
    for x in L:
        e = e + x**2
    e = e/len(L)
    e = e - moyenne(L)**2
    e = np.sqrt(e)
    return(e)
```

Entrée [5]:
```
ecarttype(notes)
```

Out [5]:
```
2.3057898140695006
```

③ Ecrire une fonction `etendue` prenant en argument une liste `L` et qui renvoie l'**étendue** de cette liste. *On indique que l'étendue des valeurs $(x_k)_{k=1,\dots,n}$ est la différence entre la plus grande valeur et la plus petite valeur parmi ces données.*

Entrée [6]:
```
def etendue(L):
    min = L[0]
    max = L[0]
    for x in L:
        if x > max :
            max=x
        if x < min :
            min = x
    return(max - min)
```

Entrée [7]:
```
etendue(notes)
```

Out [7]:
```
10
```

III Annexe : Documentation de pandas

```
import pandas as pd              # pandas : gestion de données
```

Voici une liste de commandes utiles pour ce TP. Le tableau des données importées sera ici noté `df` (dataframe) et `Colonne` est le *nom* d'une colonne de ce tableau.
Attention : adapter le df en fonction du nom donné à votre tableau au moment de l'import.

`df = pd.read_csv(fichier)`	import d'un fichier csv.
`df = pd.read_excel(fichier)`	import d'un fichier excel
`df`	aperçu du tableau
`df.head(), df.head(n)`	5 premières lignes, n premières lignes
`df.tail() , df.tail(n)`	5 dernières lignes, n dernières lignes
`df.shape`	taille du tableau (lignes, colonnes)
`df.info()`	affiche les colonnes et leur type
`df[Colonne]`	données de la colonne `Colonne`. Attention, si le nom de la colonne est une chaîne de caractères, on écrira son nom entre guillemets.
`df[[Colonne1,Colonne2,...]]`	sélection des colonnes indiquées.
`df[ df[Colonne] == 5 ]`	applique un filtre, ici la valeur de `Colonne` doit être égale à 5.
`df[ (filtre1) & (filtre2) ]`	applique plusieurs filtres (booléens), le `&` signifie 'et'
`df.sort_values(Colonne)`	trie la Colonne par ordre croissant.
`df.describe()`	statistiques de base
`df.mean()`	moyenne
`df.std()`	écart-type
`df.count()`	nombre de valeurs
`df.median()`	médiane
`df.max()`	maximum
`df.min()`	minimum
`df.sum()`	somme des données
`df.hist(column = Colonne)`	histogrammes de Colonne. Options : `range = (a,b)` (intervalle $[a, b]$ en abscisse), `bins = n` (nombre de classes)
`df.hist(column = Colonne1, by = Colonne2)`	histogrammes de Colonne1 pour chaque valeur de Colonne2.