TP 1 : Initiation à Python pour la statistique

1 Python et l'interface spyder

Python est un langage de programmation généraliste de haut niveau qui peut être téléchargé sur le site python.org et qui fonctionne sous Microsoft Windows, Mac OS et Linux.

Python dispose de nombreuses *bibliothèques* de fonctions déjà programmées, notamment :

— NumPy pour l'analyse numérique, en particulier le traitement de matrices et de tableaux de données multidimensionnels, ainsi que les constantes π (`np.pi`) et e (`np.e`) et les fonctions mathématiques usuelles : racine carrée (`np.sqrt`), logarithme (`np.log`) et exponentielle (`np.exp`), fonctions trigonométriques (`np.cos`, `np.sin`, `np.acos`, etc.), partie entière inférieure (`np.floor`) ou supérieure (`np.ceil`), etc. ;

— Matplotlib pour tracer des graphiques ;

— Pandas pour importer, explorer et traiter des données (tutoriel).

Pour travailler avec Python, il est commode d'utiliser l'interface Spyder, qui est installée sur les ordinateurs de l'université, et dont une version *open source* est téléchargeable à l'adresse `https://www.spyder-ide.org`. Si vous l'utilisez en Analyse numérique, vous pouvez aussi utiliser un Notebook.

Cette interface comporte :

— une console (*shell*), permettant d'exécuter des commandes et des scripts Python (en bas à droite),

— un éditeur permettant d'écrire des scripts Python (colonne de gauche), que l'on peut ensuite exécuter dans la console (toujours en bas à droite),

— un espace en haut à droite qui permet d'afficher en particulier un navigateur de fichiers et une fenêtre d'aide (à utiliser sans modération).

2 Prise en main

Ouvrir un nouveau *script* (fichier contenant des commandes Python) et l'enregistrer en utilisant un nom se terminant par `.py`. Taper une commande dans le script, par exemple `1+2`. Pour exécuter le script dans la console, utiliser la touche `F5`, ou la touche `F9` pour exécuter la ligne courante ou la sélection.

Pour obtenir de l'aide sur une fonction, par exemple `print`, on peut utiliser directement le moteur de recherche situé dans la fenêtre d'aide, ou taper dans la console `?print` ou `help(print)`.

Deux mises en garde : affichage des résultats et importation des bibliothèques

Mise en garde. *Si on écrit une commande dans un script, les calculs sont exécutés mais pas affichés si on ne le demande pas expressément. Par exemple, taper la commande suivante dans un script et l'exécuter (*`F9` *pour exécuter), puis faire de même (*`Entrée` *pour exécuter) dans la console :*

```
5+2
```

Faire de même avec la commande suivante et observer ce qui se passe.

```
print(5+2)
```

À présent, taper dans un script ou dans la console la commande `exp(2)` : on constate que la fonction `exp` n'est pas définie par défaut.

Mise en garde. *Il faut donc importer la bibliothèque* `numpy` *– une fois par session, inutile de le faire à chaque calcul. On importe* `matplotlib` *et* `pandas` *en même temps, ce qui conduit à écrire dans tous les scripts le préambule suivant.*

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pan
```

Écrire dans un script et lancer les commandes suivantes. Interpréter les résultats.

Commandes de base

```
print(0.1*4)
a = np.exp(2)
print(a)
# ceci est un commentaire
b = np.cos(1); print("b = %s" % b)
# remplacement de %s
# par la chaine qui suit le signe %
```

Création de tableaux NumPy

```
# une liste Python
u = [1, 9, -4, 0.5]
print(2*u)
# un tableau NumPy
v = np.array([1, 9, -4, 0.5])
print(v[2])
print(v[0])
# la numerotation commence a zero

print(v.size)
print(np.size(v))
print(v.shape)
```

Intervalles et répétitions

```
print(np.linspace(0, 2 * np.pi, 10))
print(np.linspace(20, 1, 10))
x,dx=np.linspace(0,6.3,10,retstep=True)
print(x)
print(dx)

x = np.arange(0, 10, 0.5)
print(x)

dx = 0.5
x = np.arange(0, 10+dx/2, dx)
print(x)

y = np.arange(10, -dx/2, -dx)
print(y)

print(np.repeat(3.2, 4))
x = np.array([[1,2],[3,4]])
print(np.repeat(x, 2))
print(np.repeat(x, 3, axis=1))
print(np.repeat(x, 4, axis=0))
```

Opérations sur les vecteurs

```
A = np.array([1, 2, 3, 6])
B = np.array([0, -4, 9, 4]);
print(np.exp(A))
print(A+B)
print(A*B)
print(A/B)
print(2*A)
print(A+1)
print(A+np.array([1,2,3]))
```

Sélection

```
print(A==2)
print(A[A>2])
```

Sommes et sommes cumulées

```
u = np.arange(1,10,1)
print(np.sum(u))
print(np.cumsum(u))
```

Création et opération sur les matrices

```
A.shape = (2,2); print(A)
print(A.transpose())
C = B.reshape(2,2).T
print(C); print(A @ C)
print(np.matmul(A,C))
print(C[1,]); print(C/A[1,])
```

Valeurs et vecteurs propres

La commande `np.linalg.eig(A)` permet de calculer les valeurs propres et les vecteurs propres de la matrice `A`; on obtient les valeurs propres seules avec `np.linalg.eigvals(A)`.

Tracés de graphiques

Pour tracer le graphique d'une fonction mathématique, on utilise la bibliothèque `matplotlib`, dont la partie pertinent `pyplot` a été importée sous le petit nom de `plt`.

```
x = np.linspace(0, 2*np.pi, 100)
plt.plot(x, np.sin(x))
plt.show()

plt.plot(np.cos(x), np.sin(x))
plt.show()

plt.plot(x, np.cos(x), x, np.sin(x))
plt.show()

plt.plot(x, np.cos(x))
plt.plot(x, np.sin(x))
plt.show()
```

On constate que tous les graphes sont tracés dans la même fenêtre, qui n'apparaît qu'à l'exécution de la commande `plt.show()`. Voici comment ouvrir plusieurs fenêtres.

```
x = np.linspace(0, 2*np.pi, 100)
y = np.linspace(0., 1., 100)

plt.figure(1)
```

```
plt.plot(x, np.sin(x))

plt.figure(2)
plt.plot(y, y * np.sin(y))

plt.show()

plt.plot([0, 1], [1, 0], "b")
plt.plot(0.1, 0.9, "r+")
plt.xlabel("x")
plt.ylabel("y")
plt.title("joli graphe")
plt.legend(["segment", "point"])
plt.show()
```

3 Statistique descriptive

Numpy est statisticien : `https://docs.scipy.org/doc/numpy/reference/routines.statistics.html`.

3.1 Importation de données

Pour illustrer les outils basiques de statistiques descriptives, on va charger un jeu de données, en utilisant la commande suivante.

```
df = pan.read_csv("https://tinyurl.com/donnees-YD-csv",sep="\t")
print(df)#Pour afficher, on peut aussi cliquer sur df dans l'explorateur de variables.
```

Les données correspondent à l'âge, au poids, à la taille, à la consommation hebdomadaire d'alcool (en nombre de verres bus), au sexe, au ronflement et au tabagisme, d'un échantillon de cent personnes.

On récupère (par exemple) les données concernant l'âge et le ronflement, de la façon suivante.

```
age = df["age"];
ronfle = df["ronfle"]
```

On peut aussi faire une petite boucle pour nommer toutes les variables.

```
print("Cles du tableau de donnees : %s" % df.keys())
for nom in df.keys():
    globals()[nom] = df[nom]
```

Commencer par identifier les variables qualitatives nominales, ordinales et quantitative discrètes et continues.

Remarque. *Les deux types fondamentaux de* `pandas` *sont le DataFrame (*`df` *en est un) et la Series (*`df["age"]`*, etc., en sont). Un DataFrame est un tableau de données avec des colonnes aux noms arbitraires représentant des variables de n'importe quel type. Une Series est une colonne d'un DataFrame, c'est-à-dire une variable statistique.*

3.2 Résumés numériques

En utilisant les fonctions `mean`, `var`, `quantile` (sous la forme `taille.mean()`, etc.), déterminer la moyenne empirique, la variance empirique, la variance empirique non-biaisée et l'écart-type empirique pour les variables quantitatives. Déterminer la médiane pour chaque caractéristique ordinale ou quantitative de l'échantillon.

Représenter également les données à l'aide d'un diagramme en boîte `boxplot` (sous forme `df.boxplot()` ou `df[["age"]].boxplot()`, etc.). Rappeler les éléments représentés dans ce diagramme.

3.3 Résumés graphiques

— Représenter les variables quantitatives discrètes et continues à l'aide respectivement d'un diagramme en bâtons et d'un histogramme (fonction `hist`, que l'on invoque par `df[["age","taille"]].hist()` ou `df["age"].plot.hist()`, etc.). Représenter leurs fonctions de répartitions empiriques. (On peut utiliser l'option `cumulative=True` de la fonction `hist`.)

— Représenter les variables qualitatives par un diagramme en tuyaux d'orgue :

```
C = pan.Series.value_counts(df["sexe"])
plt.bar(["Homme","Femme"],list(C),width=0.4)
```

— Donner les tables de contingences pour les variables sexe et tabagisme :

```
pan.crosstab(df["sexe"],df["alcool"])
```

Représenter par un nuage de point les variables poids et taille, par exemple par

```
pan.plotting.scatter_matrix(df[["poids","taille"]])
```

4 Lois usuelles discrètes

4.1 Importation de Scipy

Les lois usuelles sont dans la bibliothèque SciPy et plus précisément dans le paquetage stats. On les importe :

```
from scipy.stats import bernoulli, binom, geom, poisson
```

4.2 Diagrammes en bâtons

Pour tracer le diagramme en bâtons décrivant de la loi binomiale $\mathcal{B}(10, 0.25)$, on peut utiliser la fonction `pmf` (*probability mass function*).

```
n, p = 10, 0.25
ProbasB=binom.pmf(range(n+1),n,p)# probabilites de la distribution binomiale
plt.bar(range(n+1),ProbasB,width=0.05)
plt.show()
```

Tracer de même un diagramme en bâtons des lois $\mathcal{B}(10, 0.5)$, $\mathcal{B}(100, 0.25)$, $\mathcal{P}(2)$, $\mathcal{P}(10)$, $\mathcal{G}(0.75)$, $\mathcal{G}(0.25)$ et de la loi uniforme sur $\{1, 2, 3, 4, 5, 6\}$. On utilisera avec profit la page d'aide de SciPy et on fera attention à la définition de la loi géométrique $\mathcal{G}(p)$, qui pour support $\mathbb{N}^*$ dans le cours.

4.3 Fonctions de répartition

Tracer la fonction de répartition de la loi $\mathcal{B}(10, 0.25)$ (on pourra remplacer la fonction `pmf` par `cdf` (*cumulative distribution function*) et utiliser `step` au lieu de `bar` ou superposer les deux ainsi.

```
n, p = 10, .25
x= range(n+1)
figB,axB = plt.subplots()
axB.bar(x,binom.pmf(x,n,p),width=0.05)
ax2=axB.twinx()#ajoute un axe de droite
ax2.tick_params(axis="y", labelcolor="r")
ax2.set_ylim(0,1)
ax2.step(x,binom.cdf(x,n,p), where="post",color="r")#trace la fonction de repartition
plt.show()
```

Tracer de même la fonction de répartition de la loi binomiale $\mathcal{B}(100, 0.5)$ et de la loi de Poisson $\mathcal{P}(3)$. Que valent $\mathbb{P}(X \leq 3)$ et $\mathbb{P}(X > 30)$ si X suit la loi $\mathcal{B}(50, 0.2)$?

Que donne la fonction `ppf` d'une loi ? (Indication : `ppf` vient de *percent point function*.) Quel est le lien entre les fonctions `cdf` et `ppf` ?

4.4 Lien entre loi binomiale et loi de Poisson

Nous allons illustrer un théorème de convergence de la loi binomiale vers la loi de Poisson (vue en cours avant les vacances). Pour cela, représenter sur un même graphique les lois $\mathcal{B}(50, 0.2)$, $\mathcal{B}(250, 0.04)$, $\mathcal{B}(500, 0.02)$ et $\mathcal{P}(10)$.