République du Sénégal
Un Peuple – Un But – Une Foi

-*-*-*-*-*-*-*-

Ministère de l'Enseignement Supérieur et de la Recherche et de l'Innovation

UNIVERSITE DE THIES
UFR SES/ UFR SET

Master Science des Données et Applications

Travail Pratique de SDD4142-Statistiques

Auteurs :

Ousmane DIA

Abdoulaye Bara DIAW

Ndeye Fatou DIAW

Professeur :

M GNINGUE

REPONSES AUX QUESTIONS

Question 1 :

Importations des modules :

```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
```

1 Loi Binomiale $\mathcal{B}(30, 0.2)$:

```
n, p, N = 30, 0.2, 10000
B = np.random.binomial(n, p, N)
f = sps.binom.pmf(np.arange(n+1), n, p)
plt.hist(B,bins=n+1,normed=1,range=(5,n+.5),\
   color="blue",label="loi empirique")
plt.stem(np.arange(n+1),f,"r",label="loi theorique")
plt.legend()
```

2 Loi Normale $\mathcal{N}$ (3, 4):

```
mu = 3
sigma = 4
data = np.random.randn(10000) * sigma + mu
hx, hy, _ = plt.hist(data, bins=50, normed=0,color="green")
plt.ylim(0.0,max(hx)+0.05)
plt.title('Generer des valeurs aléatoires depuis depuis une loi normale avec
python')
plt.grid()
plt.savefig("numpy_random_numbers_normal_distribution.png",
bbox_inches='tight')
plt.show()
```

3 Loi de Gamma $\mathcal{Y}(10, 5)$:

```
# Nombre de valeurs à générer
length = 10000
bins=1000
###Données
data = np.random.gamma(10,5,length)
###Graphe
y, x = np.histogram(data, bins=bins, density=True)
# Milieu de chaque classe
x = (x + np.roll(x, -1))[:-1] / 2.0
plt.figure(figsize=(12,8))
plt.hist(data, bins=1000, density=True)
```

plt.title("Simulation de la loi Gamma")
plt.show()

Question 2 :

Importations des modules :

import matplotlib.pyplot as plt
import pandas as pd

1_La première étape est d'obtenir les données. Enregistrer-les dans un format adapté pour une lecture par la suite avec Python.

Réponse: *tab2 = pd.read_excel("data2.xlsx")*

2_Représentez les yi en fonction des xi. A la vue de cette représentation, pouvons-nous soupçonner une liaison linéaire entre ces deux variables ?

Réponse: *plt.scatter(x, y)*
plt.show()
Oui nous pouvons soupçonner une liaison linéaire entre les deux variables Xi et Yi.

3_Déterminer pour ces observations la droite des moindres carrés, c'est-à-dire donner les coefficients de la droite des moindres carrés.

Réponse: *Les coefficients de la droite des moindres carrées y = ax +b sont:*
a = 2.7347560975609753
b = 1.0213414634146432

4_Donner les ordonnées des yi calculés par la droite des moindres carrés correspondant aux différentes valeurs des xi.

Réponse:

x	*18*	*7*	*14*	*31*	*21*	*5*	*11*	*16*	*26*	*29*
y	*50.16*	*20.13*	*39.2*	*85.65*	*58.4*	*14.7*	*31.1*	*44.7*	*72*	*80.2*

5_Tracer ensuite la droite sur le même graphique.

Réponse :

*droite = a * x + b*
plt.plot(x, droite, "orange", x, y, "o",)
plt.show()

6_Quelle est une estimation plausible de Y à xi =21 ?

Réponse :

xi	21
y^	**58.35**

7_Quel est l'écart entre la valeur observée de Y à xi = 21 et la valeur estimée avec la droite des moindres carrés ? Comment appelons-nous cet écart ?

Réponse : *L'écart entre la valeur observée de Y à xi = 21 et la valeur estimée avec la droite des moindres carrés est : 62 - 58.35 = 3.65. Cet écart est appelé : erreur d'estimation.*

8_Est-ce que la droite des moindres carrés obtenue en 2 passe par le point (¯ x, ¯ y) ? Pouvons-nous généraliser cette conclusion à n'importe laquelle droite de régression ?

Réponse : *Oui la droite des moindres carrés obtenue en 2. passe par le point (¯ x, ¯ y). Oui nous pouvons généraliser cette conclusion à n'importe quelle droite de régression.*

Question 3 :

Importation des modules :

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
```

1_ Enregistrer les données dans un format adapté pour une lecture par la suite avec Python sachant que la première ligne du fichier smp.csv correspond aux noms des variables. Vérifier si vous avez une structure de 799 observations et 26 variables.

Réponse :

```
fichier = pd.read_excel("data3.xlsx")
print(q3.shape)
```

2_ Changer les types des variables.

Réponse :

```
q3["prof"] = q3["prof"].astype("category")
q3["duree"] = q3["duree"].astype("category")
q3["discip"] = q3["discip"].astype("category")
q3["ecole"] = q3["ecole"].astype("category")
q3["separation"] = q3["separation"].astype("category")
q3["juge.enfant"] = q3["juge.enfant"].astype("category")
q3["place"] = q3["place"].astype("category")
q3["abus"] = q3["abus"].astype("category")
q3["grav.cons"] = q3["grav.cons"].astype("category")
q3["dep.cons"] = q3["dep.cons"].astype("category")
q3["ago.cons"] = q3["ago.cons"].astype("category")
q3["ptsd.cons"] = q3["ptsd.cons"].astype("category")
q3["alc.cons"] = q3["alc.cons"].astype("category")
q3["subst.cons"] = q3["subst.cons"].astype("category")
q3["scz.cons"]= q3["scz.cons"].astype("category")
q3["char"] = q3["char"].astype("category")
q3["rs"] = q3["rs"].astype("category")
```

q3["ed"] = q3["ed"].astype("category")
q3["dr"] = q3["dr"].astype("category")
q3["suicide.hr"] = q3["suicide.hr"].astype("category")
q3["suicide.past"] = q3["suicide.past"].astype("category")

3_ Calculer la moyenne, la variance, et l'écart type pour chacune des variables suivantes : age, n.enfant, n.fratrie, dur.interv. Donner les 3 premiers quantiles pour la variable age.

Réponse :

print(age.mean())
print(age.var())
print(age.std())

print(age.describe())

4_ Tracer le boxplot pour la variable age. Quelles conclusions en tirez-vous ?

Réponse :

age.plot.box(vert = False)
plt.show()

Conclusions : - Les prisonniers de cette étude ont entre 19 et 83 ans
- 25% des prisonniers sont âgés entre 19-28ans
- 25% des prisonniers sont âgés entre 28-37ans
- 25% des prisonniers sont âgés entre 37-48ans
- 25% des prisonniers sont au moins âgés de 48ans
- Les données sont beaucoup plus dispersées dans le dernier quart(au moins âgés de 48ans) que dans les autres.

5_ Afficher les données pour les agriculteurs qui ont plus de 2 enfants.

Réponse :

d5 = q3[(enfants > 2) & (prof == "agriculteur")]
print(d5)

6_ Calculer les fréquences des modalités de la variable prof. Quelle est la catégorie modale ?

Réponse :

valeurProf = prof.values
eff = prof.value_counts()#On inclut pas les valeurs 'NaN'
total = eff.sum()
*freq = (eff/total)*100*
La catégorie modale de cette variable est 'ouvrier' avec 28.62% des prisonniers

7_ Tracer le diagramme circulaire de la variable profession

Réponse :

fig = px.pie(modal, values = 'Nombres', names = 'Fonctions', title = "Répartition des professions des prisonniers de cette étude")
fig.show()

8_ Donner les moyennes des âges par profession

Réponse :

```
print(newq3.groupby(["prof"]).mean())
```

9_ Donner la table des effectifs pour les variables prof incluant les ”NaN”.

Réponse :

```
print("\n", prof.value_counts(dropna = False))
```

10_ Donner le nombre de ”Nan” pour chaque variable.

Réponse :

```
print(q3.isnull().sum())
```

11_ Supprimer toutes les lignes contenant des ”Nan”.

Réponse :

```
cleaned = q3.dropna( axis = 0)
print(cleaned)
```

12_ Tracer l’histogramme et la densité de la variable age sur la même figure.

Réponse :

```
p = sns.distplot(cleaned["age"], bins = 20, color = "blue", kde_kws = {"color":
"orange", "lw": 2})
plt.show()
```

13_ Discrétisez la variable age. Pour ce faire on ajoutera une variable dans le DataFrame des données une nouvelle variable nommée age_classe. Cette variable aura 4 classes : [min(age), Q1],] Q1, Q2],] Q2, Q3],] Q3, max(age)]. Ou Q1, Q2, Q3 sont respectivement les 3 premiers quantiles de la variable age, min(age) et max(age) respectivement la plus petite et la plus grande valeur de la variable age.

Réponse :

```
minAge = 19
maxAge = 83
Q1 = 28
Q2 = 37
Q3 = 48
q3["age_classe"] = pd.cut(q3.age, bins = [19, 28, 37, 48, 83], labels = ["19-28",
"28-37","37-48" ,"48-83"])
```

14_ Donner les fréquences des modalités de la nouvelle variable age_classe

Réponse :

```
ageCl = q3["age_classe"]
print(ageCl.value_counts())
```

Question 4 :

Importation des modules :

```
from scipy import random
import numpy as np
import matplotlib.pyplot as plt
```

Réponse:

```
a=0
b=1
N=10000
"""Définition de la fonction"""
def func(x):
    return np.sqrt(1-(x)**2)
"""Calcule de l'aire"""
areas = []
for i in range (N):
   xrand=np.zeros(N)
   for i in range(len(xrand)):
      xrand[i] = random.uniform(a,b)
      integral =0.0
   for i in range(N):
      integral += func(xrand[i])
      answer = (b-a)/float(N)*integral
      areas.append(answer)
plt.title("Distribution d'aire avec MonteCarlo sous python")
plt.hist(areas, bin=30, ec="black")
plt.xlabel("Areas")
```

Avril 2020