Abdoulaye Djibril Ba

Alioune Cissé

Yaye Sala Touré

Travail Pratique de Statistiques

Réponses aux questions

Question 1

1- Simulation loi binomiale et histogramme

```
import numpy as np

import scipy.stats as sps

import matplotlib.pyplot as plt

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=(-0.5, n + 0.5), color = "grey")

plt.stem(np.arange(n+1), f, "r")

plt.title("Loi Binomiale")
```

2- Simulation Loi Normale et densité

```
import numpy as np

import scipy.stats as sps

import matplotlib.pyplot as plt

mu, sigma, N = 3, 4, 10000

B= np.random.randn(N) * mu + sigma

x= np.linspace(-16, 16, 10000)

y= sps.norm.pdf(x, mu, sigma)
```

```
plt.plot(x, y, color="red")

plt.grid()

plt.title("Loi Normale")
```

3- Simulation Loi Gamma et densité

```
import numpy as np

import scipy.stats as sps

import matplotlib.pyplot as plt

a, b, N = 10, 5, 10000

B= np.random.gamma(a, b, N)

x= np.linspace(-16, 20, 10000)

y= sps.gamma.pdf(x, a, b)

plt.plot(x, y, color="coral")

plt.grid()

plt.title("Loi Gamma")
```

Question 2

```
import matplotlib.pyplot as plt

from scipy import array
```

Enregistrement des données sur un format adapté

```
xi= array([18, 7, 14, 31, 21, 5, 11, 16, 26, 29])

yi= array([55, 17, 36, 85, 62, 18, 33, 41, 63, 87])
```

Représentation de yi en fonction de xi

```
plt.scatter(xi, yi)
```

Oui nous pouvons dire qu'il existe un soupçon de liaison linéaire entre ces deux variables.

Calcul de la moyenne de X et de Y

```
moyenneX = xi.mean()

moyenneY = yi.mean()
```

Calcul de la variance de X et de Y

```
varX = xi.var()

varY = yi.var()
```

Calcul de l'écart type de X et de Y

```
ecarTypeX=xi.std()

ecarTypeY=yi.std()
```

Calcul de la covariance XY

```
def cov(xi, yi):
    sum = 0
    for i in range(0, len(xi)):
        sum += (xi[i] - moyenneX) * (yi[i] - moyenneY)
    return sum/(len(xi))

CovXY = cov(xi, yi)
```

Détermination du coef de corrélation

```
r = CovXY / (ecarTypeX * ecarTypeY)
```

Détermination de a

```
a = CovXY / varX
```

Détermination de b

```
b = moyenneY - a * moyenneX
```

Détermination des nouvelles valeurs estimées de Y

```
for k in range(0, len(xi)):
    valY=(a * xi[k]) + b
    print(valY)
```

Valeurs de yi estimées

50.2469512195122

20.164634146341466

39.3079268292683

85.79878048780489

58.451219512195124

14.695121951219514

31.103658536585368

44.77743902439025

72.125

80.32926829268294

Répresentation graphique

dY= (a * xi) + b

Axes = plt.axes()

plt.xlabel("Abscisses") # nom de l'axe x

plt.ylabel("Ordonnées") # nom de l'axe y

Axes.grid() # Grille de dessin

plt.plot(xi, dY, linewidth=3, color='blue', label = "Données")#Représentation

plt.legend()# Legende du graphe

plt.title("Regression Lineaire") # Titre du graphe

plt.show()

Estimation plausible de Y à xi = 21

y21 = a * 21 + b

Détermination de l'écart

ecart = yi[4] - y21

la différence entre les points issus des données et la droite obtenue par la régression linéaire est appelée Résidus.

Oui ! La droite des moindres carrées passe le point (x,y)

<u>Question 4</u>

```
import numpy.random

import random

import math

import numpy

def integration(fonction,a,b,N):

   fonction, a, b, N = 1-x^2, 0, 1, 10000

   x = a+(b-a)*numpy.random.random_sample(N)

   p = 1.0/(b-a)

   f = fonction(x)

   moyenne = f.sum()/(N*p)

   g = f*f

   variance = g.sum()*1.0/(N*p*p)-moyenne*moyenne

   return (moyenne,math.sqrt(variance/N)*1.95)
```