Rapport : Statistique pour l'informatique

Étudiant.e(s) : N° 2105542 - **VEGA** Sara | N° 12111020 - **BOUHEDDADJ** Ahmed

MAT2071L

Université Claude Bernard Lyon 1

- Département informatique -

20/12/2022

Formation : Licence 2 Informatique

- 2022/2023 -

(Sujet : N° 79899)

Exercice 1. Statistiques

A.

1. Le type de la variable statistique est quantitative continue puisqu'il s'agisse de mesures de type **'float64'** en python, donc des **valeurs réelles**.

Code :

```
print("Le type des variables: ",Air.dtypes)
```

2. - Le nombre de jours d'observation de l'échantillon est : **731** (On a obtenu ça avec la fonction len() qui fournit la taille du tableau (+1 'Car ça commence de 0')).

- Maintenant on souhaite savoir le nombre de jours ou tous les mesures étaient prises. Pour cela, On a utilisé la fonction Air.isna() afin d'affecter une variable booléenne « False » lorsque les valeurs sont prises et une variable booléenne « True » lorsqu'il y'avait pas une prise de valeur (Dans le cas où il est marqué 'nan'). Ensuite, on a fait une boucle dans les noms des substances est la variable pour but de garder que les cellules ou les valeurs sont prise (C'est-à-dire les cellules ou il est marque « False » - En utilisant le jeu de donnée modifier avec la fonction isna), Et comme ça on a obtenu un jeu de donnée qu'avec les jours ou tous les composés était mesurés. Enfin on a calculé ça taille. **Résultat : 629**

Code :

```
AirLogic = Air.isna()

ISNA=pan.isna(Air)
for i in ['Feyzin ZI Acetylene','Feyzin ZI Benzène','Feyzin ZI Ethane','Feyzin
ZI Ethylene','Feyzin ZI Isoprene','Pierre-Bénite Acetylene','Pierre-Bénite
Benzène','Pierre-Bénite Ethane','Pierre-Bénite Ethylene','Pierre-Bénite
Isoprene']:
    ISNA=ISNA[(ISNA[i] == False)]

print("Le nombre de jours d'observation de l'échantillon: ",len(Air))
print("Le nombre de jours où tous les composés organiques volatils ont été
mesuré: ",len(ISNA))
```

3. On ajoute deux nouvelles variables booléennes **BenzèneObs**, **AutresObs** ayant pour valeur **« VRAI »** respectivement si le benzène a été observé dans toutes les stations d'observation, ou si tous les autres polluants ont été observés. On a fait ça à l'aide de deux boucles qui vérifies les conditions ci-dessus, et qui affecte les valeurs correspondantes (On commence par créer et insérer les deux nouvelles variables remplis par des 0's initialement dans le dataframe **AirLogic**)

Code :

```
AirLogic.insert(13, "BenzèneOb", 0, True)
AirLogic.insert(14, "AutresOb", 0, True)


N=len(AirLogic)

for i in range (0,N):
    if (AirLogic['Feyzin ZI Benzène'][i] == False and AirLogic['Pierre-Bénite
Benzène'][i] == False):
            AirLogic['BenzèneOb'][i] = "VRAI"
    else: AirLogic['BenzèneOb'][i] = "FAUX"

for i in range (0,N):
    if (AirLogic['Feyzin ZI Acetylene'][i] == False and AirLogicgic['Pierre-
Bénite Acetylene'][i] == False and AirLogic['Feyzin ZI Dioxyde soufre'][i] ==
False and AirLogic['Pierre-Bénite Dioxyde soufre'][i] == False and
AirLogic['Feyzin ZI Ethylene'][i] == False and AirLogic['Pierre-Bénite
Ethylene'][i] == False and AirLogic['Feyzin ZI Isoprene'][i] == False and
AirLogic['Pierre-Bénite Isoprene'][i] == False):
            AirLogic['AutresOb'][i] = "VRAI"
    else: AirLogic['AutresOb'][i] = "FAUX"
```

Après on crée un nouveau dataframe avec les deux nouvelles colonnes afin de faire leur table de **contingence** (Ici on a fait copier l'ancien dataframe **'AirLogic'** dans **'Observation'** et on a supprimer tout les colonnes Apart celle des deux variables). La table de contingence et obtenu à l'aide de la méthode **crosstab()** de la bibliothèque pandas dans python.

Code :

```
Observation = AirLogic.copy()

for i in ['Date','Feyzin ZI Acetylene','Feyzin ZI Benzène','Feyzin ZI
Ethane','Feyzin ZI Ethylene','Feyzin ZI Isoprene','Pierre-Bénite
Acetylene','Pierre-Bénite Benzène','Pierre-Bénite Ethane','Pierre-Bénite
Ethylene','Pierre-Bénite Isoprene','Feyzin ZI Dioxyde soufre','Pierre-Bénite
Dioxyde soufre']:
    del Observation[i]


data_crosstab = pan.crosstab(Observation['BenzèneOb'],
                             Observation['AutresOb'],
                                margins = False)
print(data_crosstab)
```

On obtient :

	AutresOb	
BenzèneOb	FAUX	VRAI
FAUX	97	0
VRAI	46	588

Donc on constate que le nombre de jours où tous les polluants sont observés est : 588 Jours.

4. Maintenant on va ignorer les jours de non observation (on utilise la méthode dropna pour faire), Après on affiche un tableau de description afin d'avoir quelques paramètres, et pour voir mieux on a fait calculer et afficher ligne par ligne tout les paramètres demandés (La moyenne empirique, variance empirique, variance empirique non-biaisée et le quartile à 75 % de la mesure du benzène à Feyzin)

Code :

```
JoursObsB=Air.dropna(subset=['Feyzin ZI Benzène','Pierre-Bénite Benzène'])

Resume=JoursObsB.describe()

print("La moyenne: ", np.mean(JoursObsB['Feyzin ZI Benzène']))#moyenne
print("La variance empirique: ", np.var(JoursObsB['Feyzin ZI
Benzène']))#variance empirique
print("La variance empirique non-biaisé: ", np.var(JoursObsB['Feyzin ZI
Benzène'],ddof=1))#variance empirique non biaisée
print("L'écart type': ", np.std(JoursObsB['Feyzin ZI Benzène'],ddof=1))#écart-
type
print("La médiane", np.median(JoursObsB['Feyzin ZI Benzène']))#médiane
#np.quantile(JoursObsB['Feyzin ZI Benzène'],
[0.25,0.50.75],interpolation="lower")
np.quantile(JoursObsB['Feyzin ZI Benzène'],
[0.25,0.5,0.75],interpolation="lower")
```

Résultats : (Mesures du benzène à Feyzin bien sûr)

La moyenne : 2.197

La variance empirique : 3.818

La variance empirique non-biaisé : 3.824

L'écart type : 1.955

La médiane : 1.560

Quartile à 75% : 2.930

5. Maintenant on va Créez une variable SouffreMoyen donnant la moyenne des émissions de dioxyde de soufre sur les 2 stations dans chaque journée où les mesures ont été faites. Et pour cela on a fait créer un nouveau data frame 'Souffre' (une copie de dataframe Air) Mais cette fois-ci on fait supprimer tous les jours de non-observation du dioxyde souffre dans les deux stations, comme ça on peut calculer la moyenne (sans les contraintes de 'nan'). Après on effectue les calculs et on les stocke dans une nouvelle variable insérer dans le dataframe Souffre appelée 'SouffreMoyen' qui contient les résultats en question.

Code :

```
Souffre=Air.dropna(subset=['Feyzin ZI Dioxyde soufre','Pierre-Bénite Dioxyde soufre'])
mean = (Souffre['Feyzin ZI Dioxyde soufre'] + Souffre['Pierre-Bénite Dioxyde soufre']) / 2
Souffre['SouffreMoyen'] = mean
print("La moyenne de la variable SouffreMoyen est : ",
np.mean(Souffre["SouffreMoyen"]))
```

Comme on peut le remarquer, on a calculé même la moyenne de la variable 'SouffreMoyen', ainsi que la moyenne de chaque deux mesures (Du souffre évidement) des deux stations (Pour avoir plus de données).

Résultats :

La moyenne de la variable SouffreMoyen est : 2.715

Cette moyenne est inférieure au seuil d'information de 300 **microgrammes par mètre cube d'air**. Cela signifie que les émissions de dioxyde de soufre ont été **relativement faibles** et n'ont pas dépassé le seuil d'information de référence pendant la période considérée.

6. Afin de trouver l'intervalle de confiance pour la moyenne de la variable **SouffreMoyen**, on fait l'hypothèse que la variable en question suit une loi normale. Et donc pour cela, on a créé une fonction qui prend le paramètre **x et le alpha** qui égale à **0.05** (vue que le niveau de confiance et de **95%**), après on a appliqué les calculs nécessaire (**Comme vue en cours**) afin de calculer l'intervalle de confiance. (**Intervalle bilatéral et variance connue**)

$$\left[\overline{X} - \frac{\sigma}{\sqrt{n}} z_{\alpha/2}, \overline{X} + \frac{\sigma}{\sqrt{n}} z_{\alpha/2}\right]$$

Code :

```
def Intervalle(x,alpha=0.05):
    m=np.mean(x)
    s=np.std(x,ddof=1)
    l=len(x)
    delta=(st.t.ppf(1-alpha/2,df=l-1)*s)/np.sqrt(l)
    return (m-delta,m+delta)

Intervalle(x=Souffre['SouffreMoyen'],alpha=0.05)
```

Résultats :

(2.525, 2.904)

7. Afin de trouver les **sept mesures** qui ont des semaines sans aucune observation, On a créé **trois boucles**, la **1ére** fait parcourir toute les variables concernés (Les polluants), La **2éme** ça donne l'indice des lundis de chaque semaine et la **3éme** elle fait parcourir chaque jour de la semaine en comptant le nombre de **'nan'**. A la fin de chaque itération de la deuxième boucle on test si le nombre de nan de la semaine est égale a sept si c'est le cas alors la variable testée possède au moins une semaine **sans aucune observation** et donc c'est cette variable qu'on va la supprimer par la suite (Pour le dataframe **'dfh'**).

Code :

```
for ni in np.arange(0,12,1):
    nomD=['Feyzin ZI Acetylene','Feyzin ZI Benzène','Feyzin ZI Ethane','Feyzin
ZI Ethylene','Feyzin ZI Isoprene','Pierre-Bénite Acetylene','Pierre-Bénite
Benzène','Pierre-Bénite Ethane','Pierre-Bénite Ethylene','Pierre-Bénite
Isoprene','Feyzin ZI Dioxyde soufre','Pierre-Bénite Dioxyde soufre']
    nomVar=nomD[ni]
    SemaineNan=False
    #fin de grupes de 7 a 731
    i=3
    while i<731 and SemaineNan == False:
        nbNAN=0
        #i est lundice du lundi
        for j in range (i,i+7):
            #on compte si la valeur est nan pour chaque jour de la semaine
            m=np.isnan(Air[nomVar][j])
            if(m==True):
                nbNAN=nbNAN+1
        if nbNAN == 7:
            SemaineNan=True
        i=i+7
    print("La variable",nomVar,"a au moins une semaine sans observation? ")
    print(SemaineNan)
```

Résultats :

```
La variable Feyzin ZI Acetylene a au moins une semaine sans observation?
False
La variable Feyzin ZI Benzène a au moins une semaine sans observation?
False
La variable Feyzin ZI Ethane a au moins une semaine sans observation?
False
La variable Feyzin ZI Ethylene a au moins une semaine sans observation?
False
La variable Feyzin ZI Isoprene a au moins une semaine sans observation?
False
La variable Pierre-Bénite Acetylene a au moins une semaine sans observation?
True
La variable Pierre-Bénite Benzène a au moins une semaine sans observation?
True
La variable Pierre-Bénite Ethane a au moins une semaine sans observation?
True
La variable Pierre-Bénite Ethylene a au moins une semaine sans observation?
True
La variable Pierre-Bénite Isoprene a au moins une semaine sans observation?
True
La variable Feyzin ZI Dioxyde soufre a au moins une semaine sans observation?
True
La variable Pierre-Bénite Dioxyde soufre a au moins une semaine sans
observation?
True
```

On trouve sept variables (Mis en highlight ci-dessus) qui ont des semaines sans aucune observation.

Maintenant on va créer un nouveau dataframe **dfh** dont les 7 sept variables ci-dessous sont exclu. Pour cela On fait copier le dataframe **Air** dans **dfh** et on supprime les **7 variables** (On a fait une boucle pour ça), Tel que pour individus les semaines d'observation et pour variables : les **moyennes hebdomadaires** de

pollutions, pour chacune des **5 mesures** ayant des observations pour toutes les **semaines**. **(dfh devrai contenir 105 semaines, et 5 variables – C'est ce qu'on a trouvé)**

Code :

```
dfh = Air[['Date','Feyzin ZI Acetylene','Feyzin ZI Benzène','Feyzin ZI
Ethane','Feyzin ZI Ethylene','Feyzin ZI Isoprene']]

AirSept = Air.copy()

for i in ['Pierre-Bénite Acetylene','Pierre-Bénite Benzène','Pierre-Bénite
Ethane','Pierre-Bénite Ethylene','Pierre-Bénite Isoprene','Feyzin ZI Dioxyde
soufre','Pierre-Bénite Dioxyde soufre']:
    del AirSept[i]

dfhSept = AirSept.copy()

dfhSept.set_index('Date', inplace=True)

dfhSept.index = pan.to_datetime(dfhSept.index)

dfh = dfhSept.resample('W').mean()
```

B.

1.

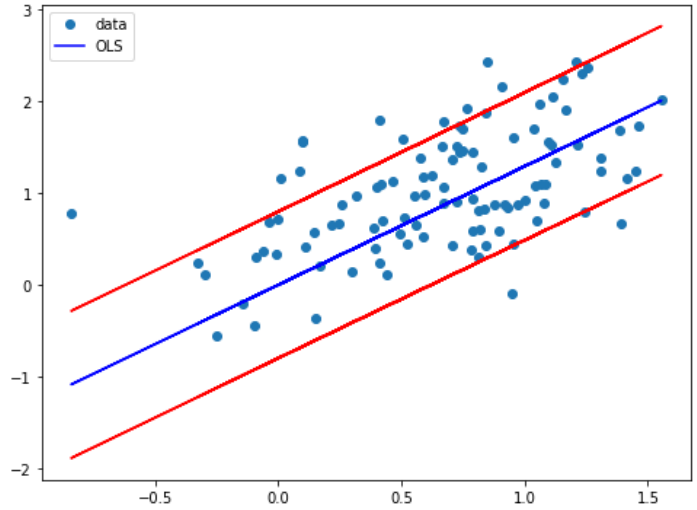
Code :

```
#On fait insérer les deux nouvelles colonnes qui contients les log des mésures
de benzene et Ethylene

dfh.insert(5, 'logarithme de la mesure du Benzène', 0, True)
dfh.insert(6, 'logarithme de la mesure de l'Éthylène', 0, True)

#On calcule le logarithme de base exponentielle des mesures (Logarithme
népérien)
N=len(dfh)
for i in range (0,N):
    dfh['logarithme de la mesure du Benzène'][i]=math.log(dfh['Feyzin ZI
Benzène'][i],math.e)
    dfh['logarithme de la mesure de l'Éthylène'][i]=math.log(dfh['Feyzin ZI
Ethylene'][i],math.e)

#On crée le model en question avec le y'mesure de l'Éthylène' et le x'mesure du
Benzène'
model = sm.OLS(dfh['logarithme de la mesure de l'Éthylène'], dfh['logarithme de
la mesure du Benzène']).fit();print(model.summary())

#On affecte les séries de valeurs aux x et y avec ce qui correspond
x = dfh['logarithme de la mesure du Benzène']
y = dfh['logarithme de la mesure de l'Éthylène']

#La partie du traçage du nuage de points avec la droite de regression avec les
deux droites donnant les bornes de l'intervalle
#de prédiction au niveau 80%
prediction=model.get_prediction().summary_frame(alpha=0.2)

fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(x, y, "o", label="data")
ax.plot(x,  prediction["mean"], label="OLS",color="blue")#droite de régression
ax.plot(x, prediction["obs_ci_lower"], color="red")#borne inf de la prédiction
ax.plot(x, prediction["obs_ci_upper"], color="red")#borne sup  de la prédiction
ax.legend(loc="best")
fig.suptitle("Régression du logarithme de la mesure de l'Éthylène en fonction
de logarithme de la mesure du Benzène avec intervalle de prédiction (rouge)")
```

Résultat :

```
#                                 OLS Regression Results
# ==============================================================================
# Dep. Variable:     logarithme de la mesure de l'Éthylène   R-squared (uncentered):                   0.743
# Model:                                               OLS   Adj. R-squared (uncentered):              0.740
# Method:                                    Least Squares   F-statistic:                              300.4
# Date:                                   Mon, 19 Dec 2022   Prob (F-statistic):                    1.95e-32
# Time:                                           14:31:38   Log-Likelihood:                         -97.841
# No. Observations:                                    105   AIC:                                      197.7
# Df Residuals:                                        104   BIC:                                      200.3
# Df Model:                                              1
# Covariance Type:                               nonrobust
# ==============================================================================
#                                  coef    std err          t      P>|t|      [0.025      0.975]
# ------------------------------------------------------------------------------
# logarithme de la mesure du Benzène     1.2889      0.074     17.331      0.000       1.141       1.436
# ==============================================================================
# Omnibus:                        0.608   Durbin-Watson:                   1.349
# Prob(Omnibus):                  0.738   Jarque-Bera (JB):                0.670
# Skew:                           0.175   Prob(JB):                        0.715
# Kurtosis:                       2.823   Cond. No.                         1.00
# ==============================================================================

# Notes:
# [1] R² is computed without centering (uncentered) since the model does not contain a constant.
# [2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Régression du logarithme de la mesure de l'Éthylène en fonction de logarithme de la mesure du Benzène avec intervalle de prédiction (rouge)



Conclusion :

On observe que : Le coefficient de détermination (R^2) est de 0,743 ce qui est proche de 1 alors cette la régression est un bon modèle pour les données.

La Probabilité noté Prob (F-statistic) est de 1.95e-32 ce qui est très proche de zéro, alors cela implique que la régression est significative

Donc ce résultat est significatif.

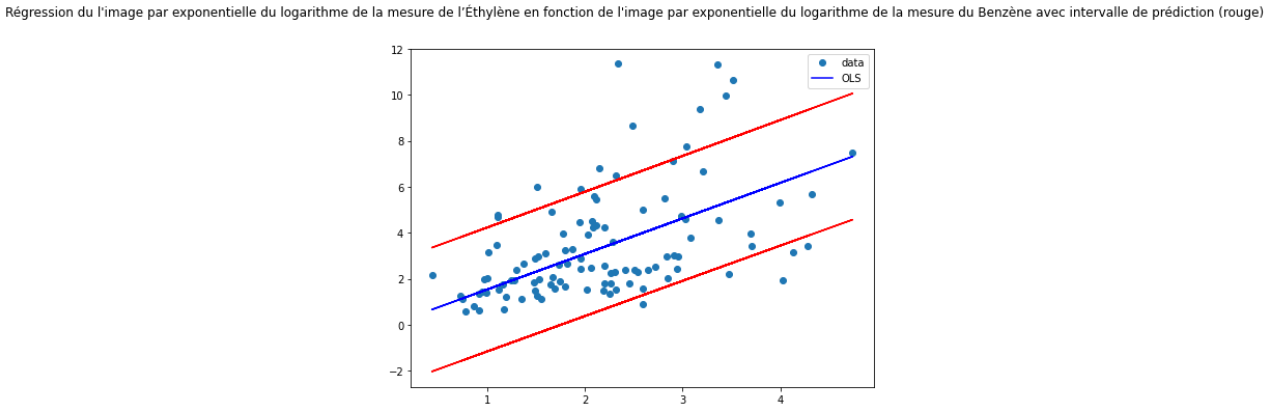
2. Maintenant, on fait pareil mais cette fois-ci on prend l'image par exponentielle de la régression linéaire de x et de y

Code :

```
#Ici on prend l'image par exponentielle de la régression linéaire de x et de y
xx= dfh['logarithme de la mesure du Benzène'].apply(lambda x: math.exp(x))
yy= dfh['logarithme de la mesure de l'Éthylène'].apply(lambda x: math.exp(x))

#On crée un nouveau model a partir des nouveau xx et yy (qui sont des image exponentielle)
model2 = sm.OLS(yy,xx).fit();print(model.summary())

prediction2=model2.get_prediction().summary_frame(alpha=0.2)

#Cette partie concerne le tracage des nuages de points ainsi que la courbe de regression
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(xx, yy, "o", label="data")
ax.plot(xx,  prediction2["mean"], label="OLS",color="blue")#droite de régression
```

```
ax.plot(xx, prediction2["obs_ci_lower"], color="red")#borne inf de la
prédiction
ax.plot(xx, prediction2["obs_ci_upper"], color="red")#borne sup  de la
prédiction
ax.legend(loc="best")
fig.suptitle("Régression du l'image par exponentielle du logarithme de la
mesure de l'Éthylène en fonction de l'image par exponentielle du logarithme de
la mesure du Benzène avec intervalle de prédiction (rouge)")
```

Résultats :



3. Pour calculer les intervalles de prédiction pour la mesure d'Éthylène si la mesure en Benzène est au seuil réglementaire d=1, On utilise la méthode get_prediction().

Code :

```
#On calcule les intervalles de prédiction a l'aide de la methode get_prediction
avec fixation benzene au seuil réglémenté d=1
model.get_prediction(exog=[1]).summary_frame(alpha=0.2)
```

Résultats:

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	1.288873	0.074368	1.192957	1.384788	0.486906	2.090839

Conclusion:

On calcule avec le code si dessous, Si la mesure de benzène est au niveau réglementaire 1, cela signifie que la concentration de benzène dans l'échantillon est inférieure à cette limite.

On a une estimation de la plage des valeurs la mesure de l'éthylèneavec la moyenne estimée, l'incertitude de cette moyenne (mean_se), et les bornes de l'intervalle de confiance (mean_ci_lower et mean_ci_upper).

Alors ces intervalles peuvent être utilisés pour évaluer la confiance que l'on peut avoir dans les estimations.

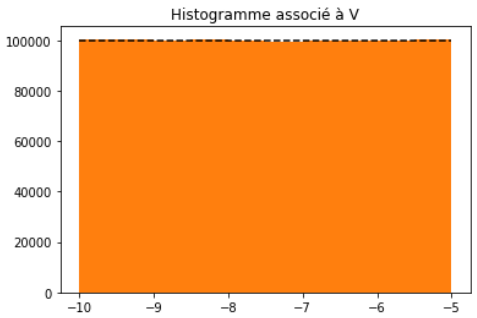
Donc le benzène peut être utilisé comme indicateur pollution puisque sa présence dans l'atmosphère est dangereuse pour la santé.

Exercice 2.

1. Code :

```
N = 1000000  # taille du vecteur U
#on cree le verteur U
U = st.uniform.rvs(-10,5, size=N)
plt.hist(U)
#on crée le vecteur V
V = U[U < 0]
plt.hist(V)
xmin, xmax = -10, -5

plt.plot([xmin, xmax], [100000, 100000], 'k--')
plt.title('Histogramme associé à V')
```

Résultats :

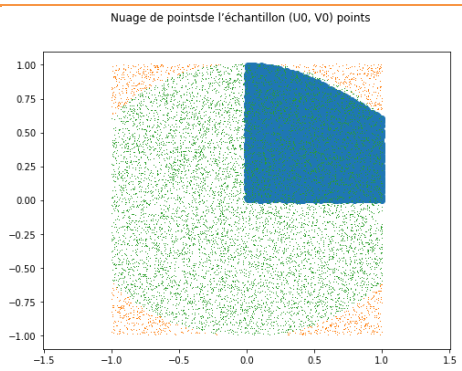


En comparant l'histogramme associé à V avec la densité d'une loi uniforme bien choisie, trouvez de quelle loi V est un échantillon. N = 1000000 nous a permis de conclure, la densité d'une loi uniforme entre -10 et -5 de densité 1/5.

2. Code :

```
#on trace le nuage de points
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(U0, V0, "o", label="points")
fig.suptitle("Nuage de pointsde l'échantillon (U0, V0) points .")

# Définir les demi-axes de l'ovale
a = 1.254
b = 1.01
#on genere les points pour tester l'ecuation
U=st.uniform.rvs(size=10000,loc=-1,scale=2)
V=st.uniform.rvs(size=10000,loc=-1,scale=2)

#c'est l'équation lour les poins dans l'aire d'une ellipse (x^2)/a^2 + (y^2)/b^2 <= 1
S=U**2/a**2+V**2/b**2  #ellipse
Cas =(S<1)  # condition
U0=U[Cas]    # elts qui verifient rayon <1
V0 = V[Cas]   # elts qui verifient rayon <1
plt.axis('equal')
plt.plot(U,V,marker=',',linestyle='')

#l'échantillon est uniformement distribué sur [-1,1][-1,1]
plt.axis('equal')
plt.plot(U0,V0,marker=',',linestyle='')

#l'echantillon du couple (U0,V0) est une ellipse de centre 0,0 de rayon 1 qui a 0<U0<1 et
0<V0<1
```

Résultats :

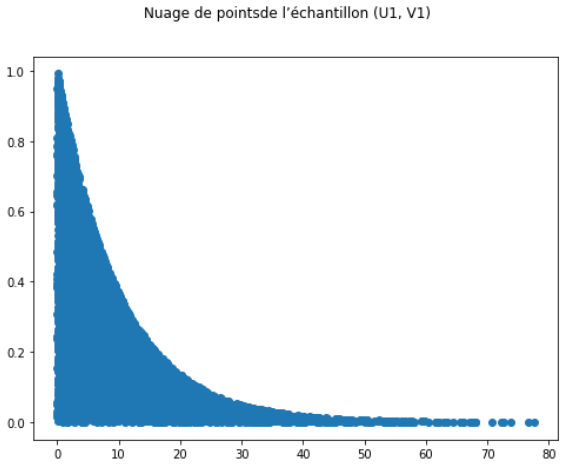


En bleu on a le nuage de points de l'échantillon (U0 ,V0).
On trouve cette figure, alors l'ensemble de R2 l'échantillon du couple (U0, V 0) est uniformément distribué est une ellipse de centre 0,0 de rayon 1 qui a 0<U0<1 et 0<V0<1.

Code :

```
N = 10000
U=st.uniform.rvs(0,1,size=N)
V=st.uniform.rvs(0,1,size=N)
U0 = U
V0 = np.exp(-U0**2/2)*V
U1=-(10*np.log(U))
V1=U*V
#on trace le nuage de points
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(U1, V1, "o", label="points")
fig.suptitle("Nuage de pointsde l'échantillon (U1, V1)")
```

Résultats :



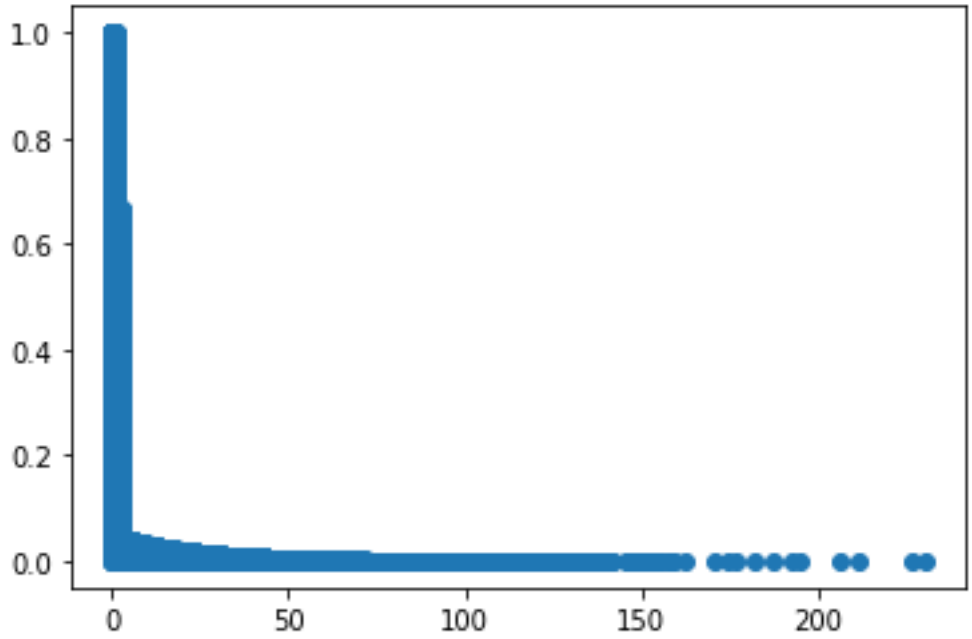
On observe que le domaine de définition de (U1, V1) est défini par l'ensemble de valeurs telles que U1 > 0 et 0 <= U < 1

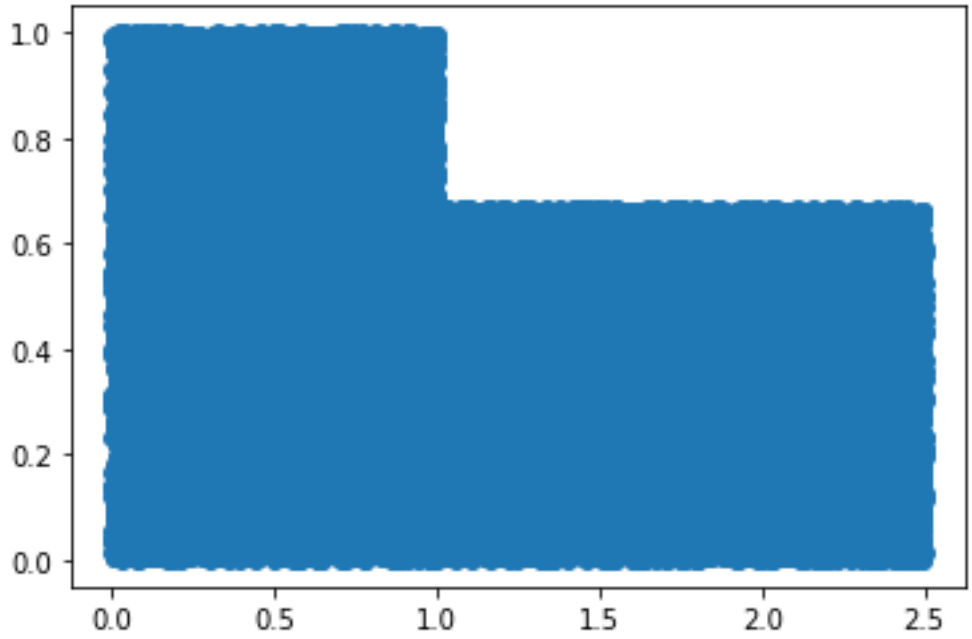
3. Code :

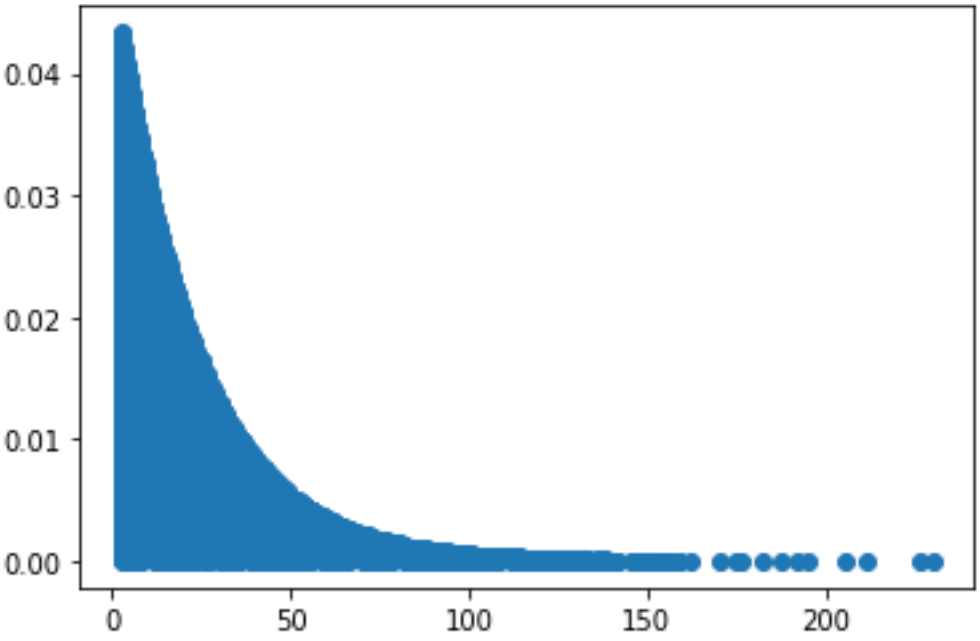
```
# Tracer le nuage de points
plt.show()
plt.scatter(T[0,], T[1,])
plt.show()

# Filtrer l'échantillon en ne conservant que les vecteurs vérifiant T[0,] < 2.5
T_below_2_5 = T[:, np.where(T[0,] < 2.5)[0]]

# Filtrer l'échantillon en ne conservant que les vecteurs vérifiant T[0,] > 2.5
T_above_2_5 = T[:, np.where(T[0,] > 2.5)[0]]

# Tracer le nuage de points pour T_below_2_5
plt.scatter(T_below_2_5[0,], T_below_2_5[1,])
plt.show()
```

```
# Tracer le nuage de points pour T_above_2_5
plt.scatter(T_above_2_5[0,], T_above_2_5[1,])
plt.show()
```

Résultats : (Les graphes sont présentés par ordre)


1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200

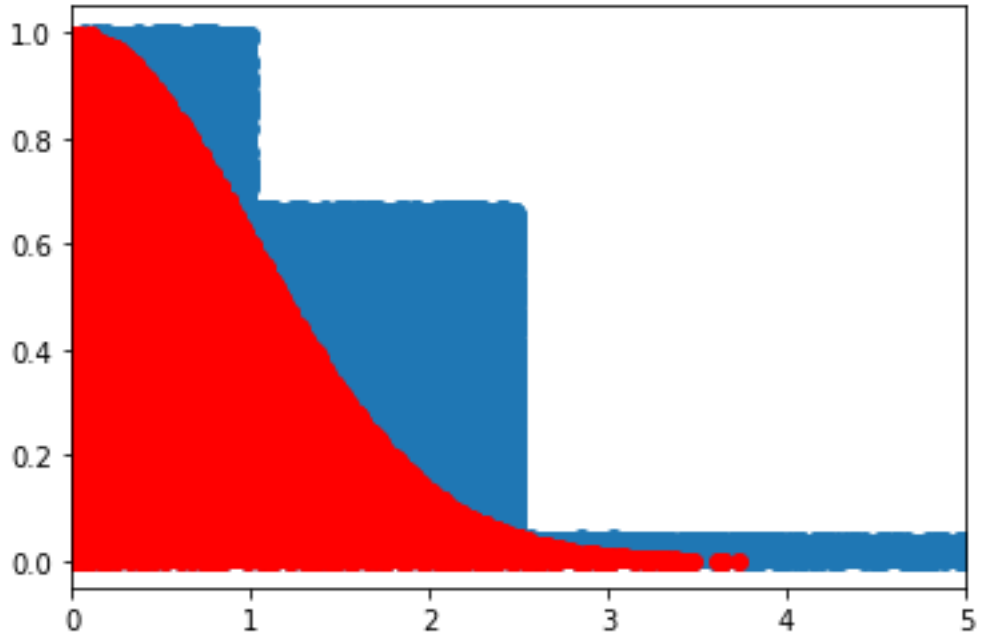

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5

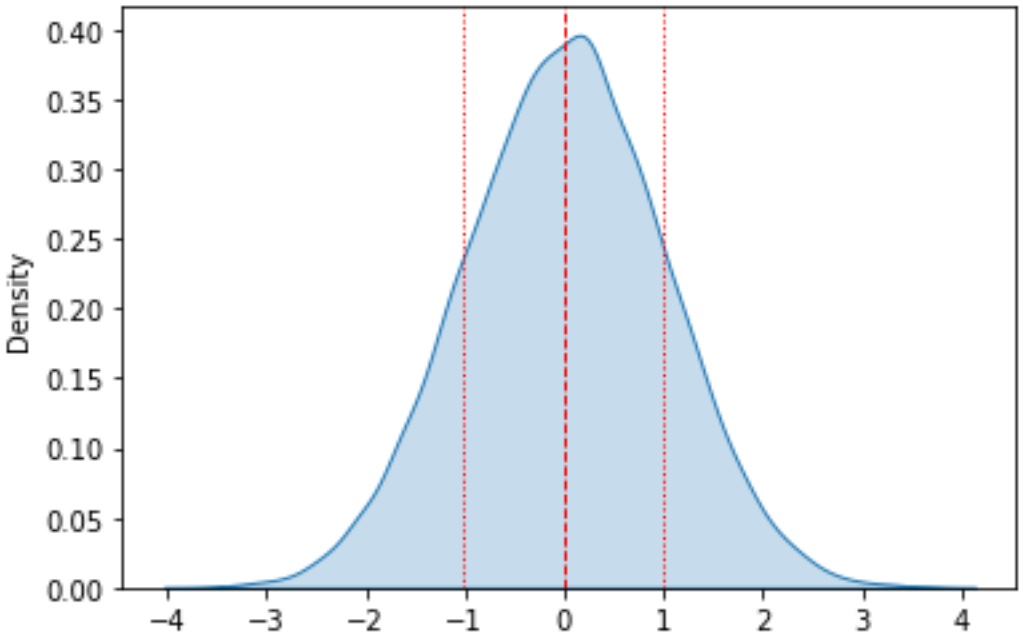

0.04
0.03
0.02
0.01
0.00
0
50
100
150
200

4. Code :

```
#Tracer le nuage de points pour T
plt.scatter(T[0,], T[1,])

# Tracer le nuage de points pour R1 et S
plt.scatter(R1, S, color='red')

# Restreindre l'axe des abscisses à [0, 5]
plt.xlim(0, 5)

plt.show()

import seaborn as sns
#On install d'abord avec %pip install seaborn

# Calculer la moyenne et l'écart-type de R2
mean_R2 = np.mean(R2)
std_R2 = np.std(R2)

# Tracer une densité de probabilité de R2
sns.kdeplot(R2, shade=True)

# Afficher la moyenne et l'écart-type de R2
plt.axvline(mean_R2, color='red', linestyle='dashed', linewidth=1)
plt.axvline(mean_R2 + std_R2, color='red', linestyle='dotted', linewidth=1)
plt.axvline(mean_R2 - std_R2, color='red', linestyle='dotted', linewidth=1)

plt.show()
```

Résultat :

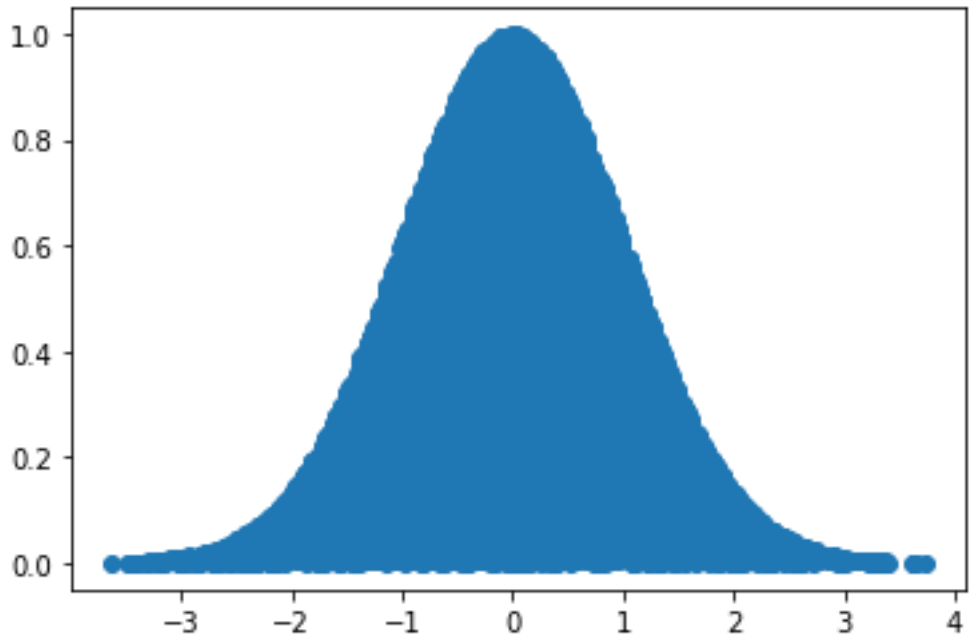


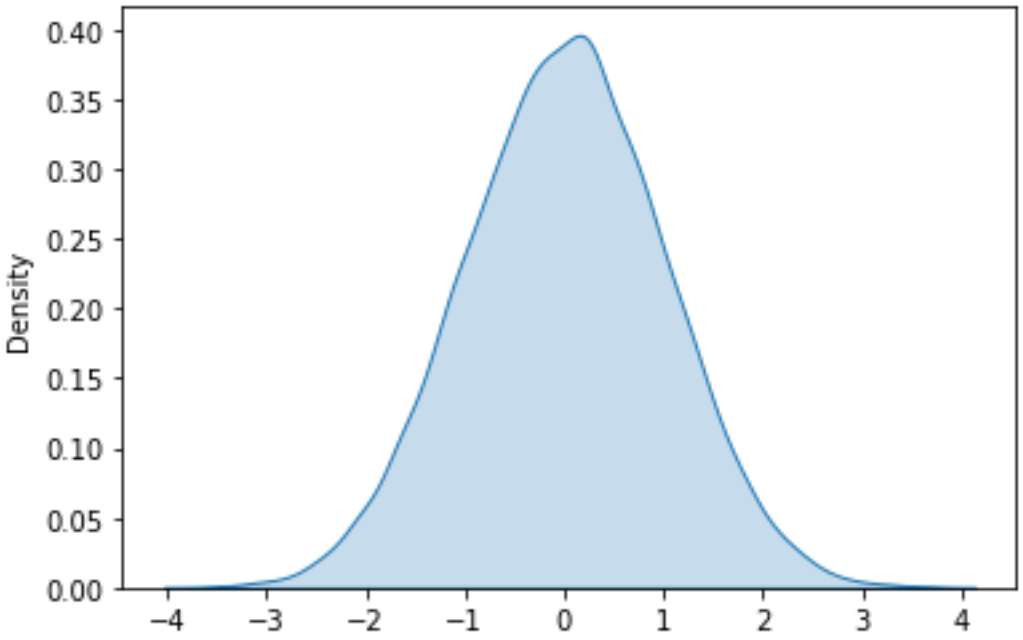


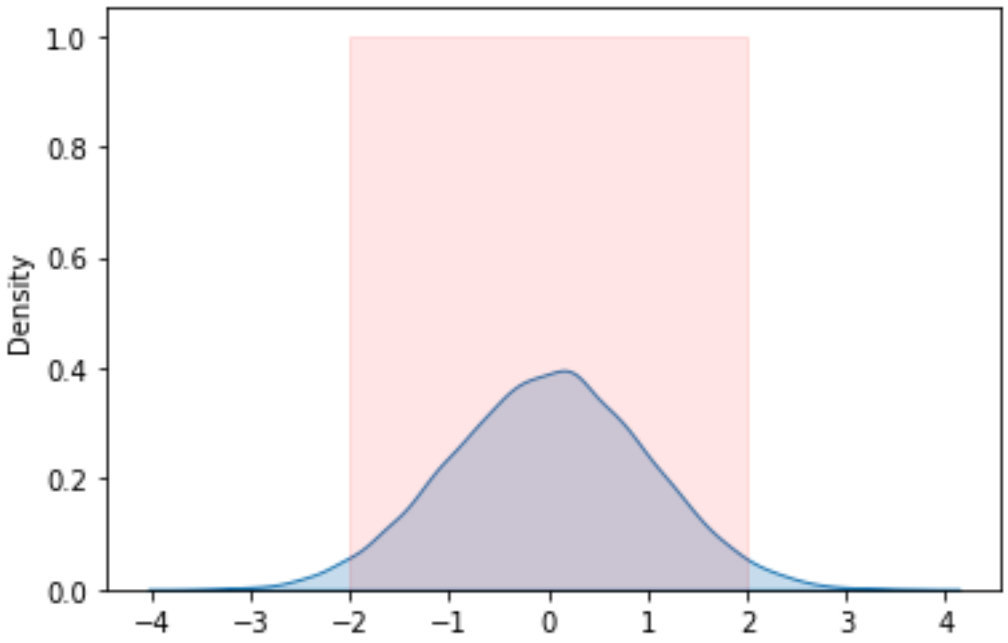
5. Code :

```
# Tracer le nuage de points pour R2 et S
plt.scatter(R2, S)

plt.show()

# Tracer une densité de probabilité de R2
sns.kdeplot(R2, shade=True)

plt.show()

# Tracer une densité de probabilité de R2
sns.kdeplot(R2, shade=True)

# Tracer une courbe du bord entre -2 et 2
plt.fill_between(x=np.arange(-2, 2, 0.01), y1=0, y2=1, color='red', alpha=0.1)

plt.show()
```

Résultats :


1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3
4


Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4


Density
1.0
0.8
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
1
2
3
4

Explication :

1. On génère un échantillon de 8 variables aléatoires uniformes Y de taille 50000.
2. On crée un échantillon de 2 variables aléatoires T de taille 50000, initialisé à 0.1.
3. On calcule la variable X en utilisant la condition 3*Y[6,]<1.
4. On calcule la variable Z en utilisant la condition 3*Y[6,]>2.
5. On met à jour la première variable T en utilisant la formule suivante : Y[0,]X + (Y[2,]3/2+1)(1-X)(1-Z) + (2.5-np.exp(25/8)*np.log(Y[4,]))*Z
6. On met à jour la deuxième variable T en utilisant la formule suivante : Y[1,]X + (Y[3]2/3)(1-X)(1-Z) + (np.exp(-25/8)*Y[4,]*Y[5,])*Z
7. On calcule la variable Cas en utilisant la condition

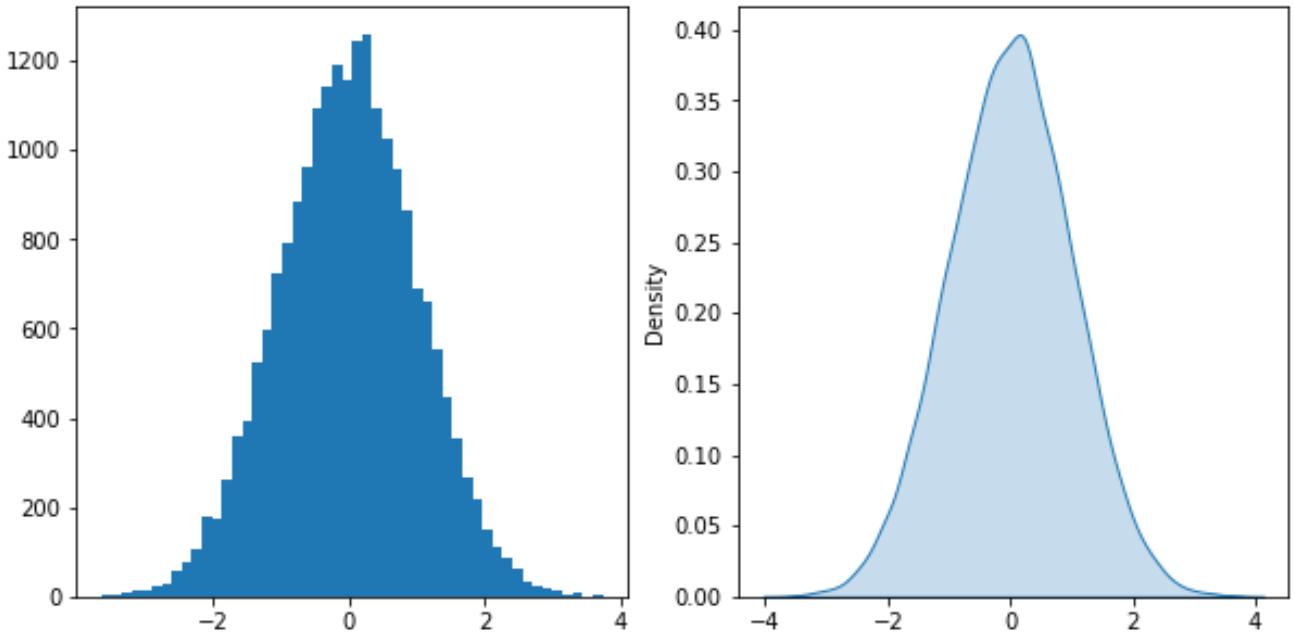
6.

Pour faire l'hypothèse, on trace côte à côte l'histogramme et la densité de probabilité.

Code :

```
# Tracer un histogramme de R2
plt.hist(R2, bins=50)

# Tracer une densité de probabilité d'une loi normale
sns.kdeplot(R2, shade=True)

# Créer une figure à deux tracés
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

# Tracer un histogramme de R2 dans le premier tracé
ax[0].hist(R2, bins=50)

# Tracer une densité de probabilité d'une loi normale dans le second tracé
sns.kdeplot(R2, shade=True, ax=ax[1])

plt.show()
```

Résultats :



Conclusion :

Il semble très similaire, on dit donc que R2 est un échantillon d'une loi uniforme continue.