L3 Informatique
TP de probabilités et statistiques

Alice Gydé et Coline Trehout

16 janvier 2022

Table des matières

Liste des tableaux

Table des figures

1 Introduction

Ce TP est divisé en trois parties. La première partie concerne la détermination des coefficients de régression linéaire par trois méthodes : la méthode de calcul des moindres carrés, la méthode matricielle et la méthode de descente de gradient. La seconde partie permet l'étude de différentes lois de probabilité : la loi binomiale, normale et exponentielle. La troisième partie concerne le calcul des intervalles de confiance sur trois exemples concrets. Les codes correspondants à chaque partie sont répartis dans les fichiers `tp1.py`, `tp2.py` et `tp3.py`.

2 Régression linéaire

La régression linéaire consiste à déterminer une droite qui réduit les écarts entre les valeurs de sortie prévues et réelles. Parmi toutes les droites possibles, on cherche la droite pour laquelle la somme des carrés des écarts verticaux des points à la droite est minimale.

Nous allons étudier l'échantillon donné dans le tableau 1 qui représente le pourcentage de rendement y_i , en fonction de la température x_i en degrés Celsius d'un procédé chimique.

x_i	45	50	55	60	65	70	75	80	85	90
y_i	43	45	48	51	55	57	59	63	66	68

TABLE 1 – Échantillon du pourcentage de rendement y_i , en fonction de la température x_i en degré Celsius d'un procédé chimique.

Il est prévu que le pourcentage du rendement d'un procédé chimique est lié linéairement à la température. L'objectif est donc d'estimer un modèle de régression linéaire simple à partir de cet échantillon. Pour cela, trois méthodes seront appliquées et comparées : la méthode de calcul des moindres carrés, la méthode matricielle et la méthode de descente de gradient.

2.1 Méthode des moindres carrés

La méthode des moindres carrés est une méthode analytique permettant de calculer les coefficients de régression linéaire. On considère le modèle de régression linéaire suivant : $Y_i = \beta_1 x + \beta_0$. Les coefficients β_0 et β_1 sont calculés grâce aux formules suivantes :

Moyenne des valeurs de x :

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Moyenne des valeurs de y :

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Coefficients de régression linéaire :

$$\beta_1 = \frac{cov(x,y)}{cov(x,x)}$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

Les points des données et la droite de régression obtenus sont tracés figure 1. La droite obtenue avec la fonction polyfit de Python superpose parfaitement celle tracée en rouge. Elle n'est donc pas affichée pour une meilleure lisibilité de la figure.

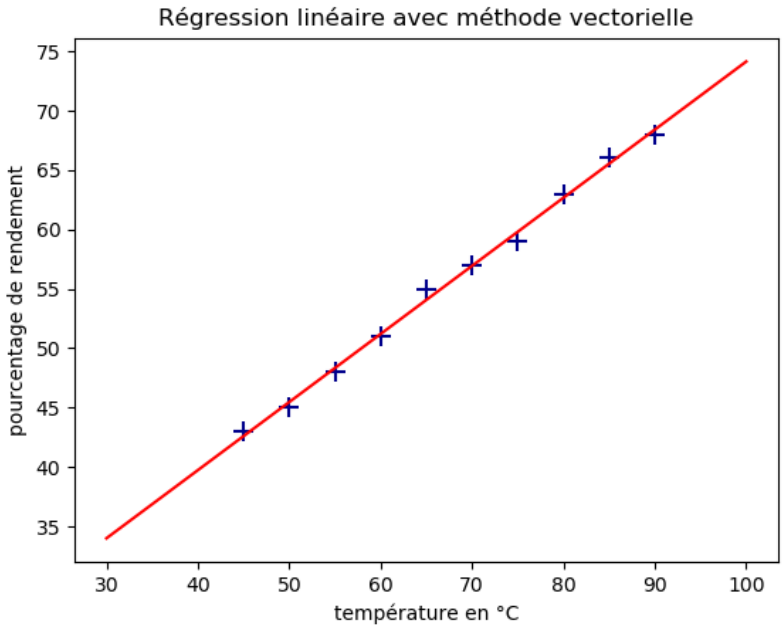

FIGURE 1 – Droite obtenue avec la méthode des moindres carrés

Les coefficients de régression obtenus sont donnés dans le tableau 2.

β_0	β_1
16.8000	0.5733

TABLE 2 – Coefficients de régression obtenus par la méthode des moindres carrés

2.2 Méthode matricielle

Cette méthode est équivalente à la méthode des moindres carrés mais le calcul se fait sous forme de matrice. Nous avons :

$$y = A\beta$$

avec :

- β le vecteur des coefficients de régressions inconnus à estimer
- A la matrice de régression
- y le vecteur des données des sorties

La matrice A a 2 colonnes et autant de lignes que de valeurs de x (ou y). La première colonne de A contient pour chaque ligne i la valeur 1 si le coefficient y_i est positif, -1 s'il est négatif et 0 s'il est nul. La deuxième colonne contient les valeurs de x. Donc avec l'exemple considéré, la matrice A ne contient que des 1 dans la première colonne. β est calculé grâce à la formule suivante :

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

$$\beta = (A^T A)^{-1} A^T Y$$

Les points des données et la droite de régression sont tracés figure 2.

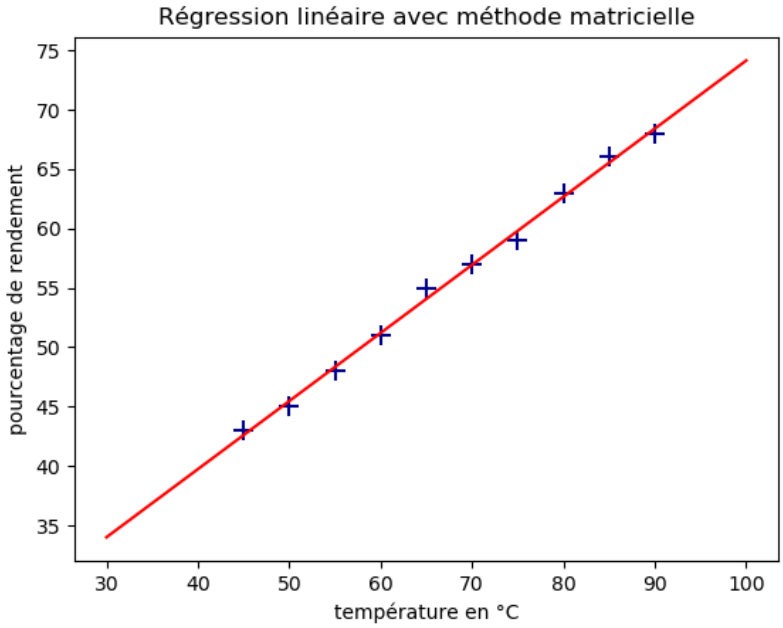

FIGURE 2 – Droite obtenue avec la méthode matricielle

Les coefficients de régression obtenus sont donnés dans le tableau 3.

β_0	β_1
16.8000	0.5733

TABLE 3 – Coefficients de régression obtenus par la méthode matricielle

Les coefficients obtenus sont égaux à ceux calculés par la méthode des moindres carrés.

L'inversion de grandes matrices étant très coûteuse, cette méthode doit être utilisée sur de petits ensembles de données seulement ou pour des matrices comportant beaucoup de zéros.

La droite du QQ-Plot qui indique la position que devraient avoir les points s'ils obéissaient exactement à la distribution normale est donnée figure 3.

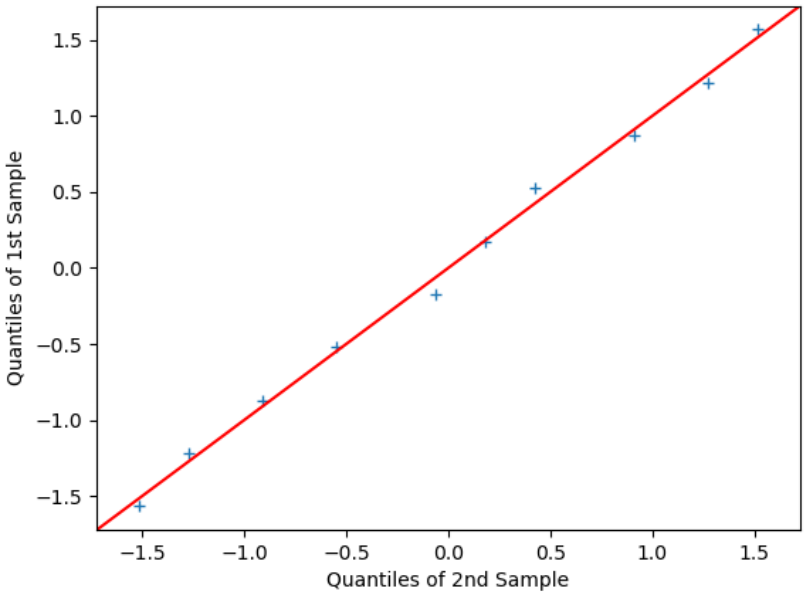

FIGURE 3 – Droite du QQ-Plot

2.3 Méthode de descente de gradient

La méthode de descente de gradient est une méthode itérative qui permet de trouver le minimum d'une fonction convexe en convergeant progressivement vers celle-ci. Dans notre cas, l'objectif est de minimiser l'écart entre les points donnés dans le tableau 1 et la droite de régression linéaire. Cette méthode donne une estimation plus ou moins bonne des paramètres de régression.

Soit β^t le vecteur des coefficients de régression à l'itération actuelle et β^{t-1} ce même vecteur à l'itération précédente. Le vecteur à l'itération actuelle est calculé grâce au précédent par la formule suivante :

$$\beta^t = \beta^{t-1} - \lambda \frac{\partial f(\beta)}{\partial \beta}$$

Les dérivées partielles par rapport à β_0 et β_1 sont calculées de la manière suivante :

$$\frac{\partial}{\partial \beta_0} f(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=0}^{n} (h(x_i) - y_i)$$

$$\frac{\partial}{\partial \beta_1} f(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=0}^{n} (h(x_i) - y_i) x_i$$

L'algorithme s'arrête lorsque $||\beta^t - \beta^{t-1}|| \leq \epsilon$.

Pour obtenir une estimation satisfaisante, nous avons fixé les paramètres suivants :

- le pas de descente $\lambda \in [0, 1]$: $\lambda = 0.0001$
- le seuil de tolérance : $\epsilon = 0.01$ ou 0.001
  plus ϵ est petit et plus la solution trouvée s'approche de la solution exacte (qui est obtenue par la méthode des moindres carrés)
- les coefficients de départ : $\beta_0 = \beta_1 = 17$

Ces valeurs donnent les meilleurs résultats par rapport aux données considérées.

Les droites obtenues par la méthode de descente de gradient pour $\epsilon = 0.01$ et $\epsilon = 0.001$ sont tracées en vert sur les figures 4 et 5. Par comparaison, la droite obtenue par la méthode des moindres carrés est tracée en rouge sur ces mêmes graphiques. Les deux droites sont très proches mais le résultat obtenu est très sensible au choix des paramètres de départ.

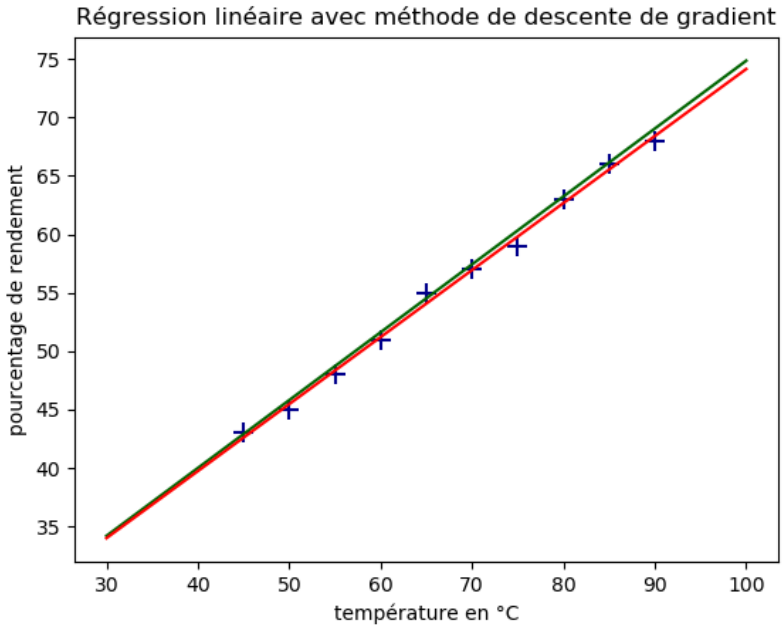

FIGURE 4 – Droite obtenue avec la méthode de descente de gradient avec $\epsilon = 0.01$

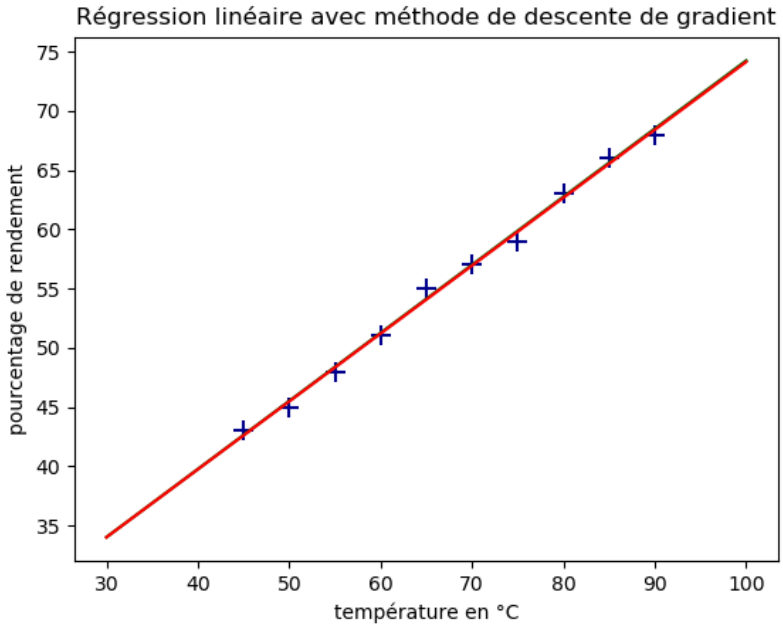

FIGURE 5 – Droite obtenue avec la méthode de descente de gradient avec $\epsilon = 0.001$

Les coefficients de régression obtenus après 12 itérations sont donnés dans le tableau 4.

β_0	β_1
16.7673	0.5808

TABLE 4 – Coefficients de régression obtenus par la méthode de descente de gradient avec $\epsilon = 0.01$

Les coefficients de régression obtenus après 15 itérations sont donnés dans le tableau 5.

β_0	β_1
16.7672	0.5748

TABLE 5 – Coefficients de régression obtenus par la méthode de descente de gradient avec $\epsilon = 0.001$

2.4 Comparaison des résultats

Les résultats obtenus avec les trois méthodes précédentes sont comparés avec ceux de la fonction polyfit de Python dans le tableau 6.

	polyfit	moindres carrés	matricielle	gradient
β_0	16.8000	16.8000	16.8000	16.7672
β_1	0.5733	0.5733	0.5733	0.5748
écart β_0	/	$7.11.10^{-15}$	$1.92.10^{-13}$	$3.28.10^{-2}$
écart β_1	/	0	$2.78.10^{-15}$	$1.47.10^{-3}$

TABLE 6 – Comparaison des coefficients de régression linéaire obtenus par les 3 méthodes avec ceux de la fonction polyfit

Pour les méthodes des moindres carrés et polyfit, l'écart est quasi nul car la fonction polyfit utilise la méthodes moindres carrés. L'écart est un peu plus grand entre la méthode matricielle et la fonction polyfit. Cela est du aux erreurs d'arrondis lors de l'inversion de la matrice. L'écart le plus grand est obtenu avec la méthode de descente de gradient car c'est une méthode approchée. Elle peut néanmoins être pertinente dans certains cas, notamment lorsque aucune méthode analytique ne peut être utilisée.

3 Étude et manipulation de lois de probabilités

3.1 Loi Binomiale

Une variable aléatoire X suit une loi binomiale de paramètre (n,p), où $n \in \mathbb{Z}_+^*$ et $0 \leq p \leq 1$, notée $\mathcal{B}(n,p)$, si $X(\Omega) = \{0,1,...,n\}$ et si sa densité de probabilité est définie par :

$$P[X=i] = \binom{n}{i} p^i (1-p)^{n-i}.$$

$$\text{Espérance} : E[X] = np$$

$$\text{Variance} : var(X) = np(1-p)$$

n étant le nombre d'expériences réalisées et p la probabilité de succès.

Les lois de probabilités de la loi binomiale sont calculées grâce à la fonction `binom.pmf`. Les lois de probabilité $\mathcal{B}(30,0.5)$, $\mathcal{B}(30,0.7)$ et $\mathcal{B}(50,0.4)$ sont représentées figure 6.

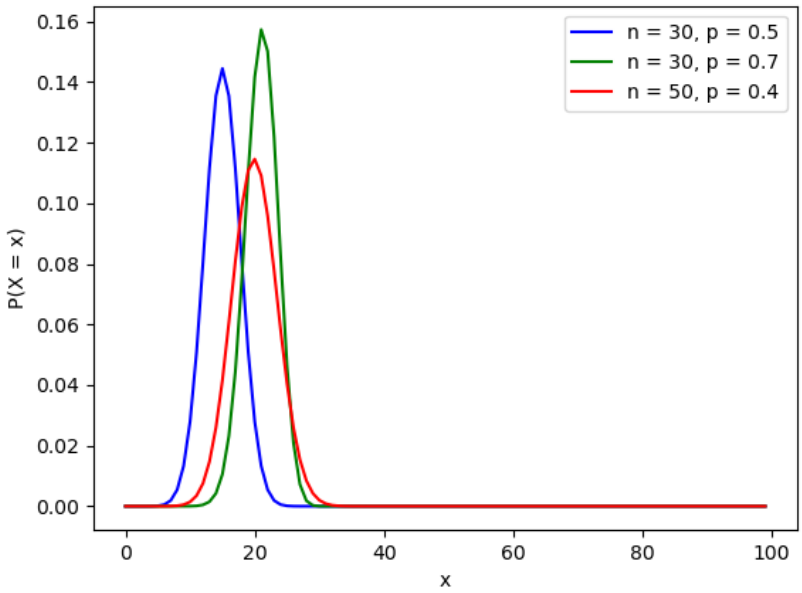

FIGURE 6 – Loi probabilité de la loi binomiale dans trois cas distincts

3.2 Loi Normale univariée

Une variable aléatoire X suit une loi normale de paramètres (m, σ) (m étant la moyenne ou espérance et σ l'écart-type) notée $\mathcal{N}(m, \sigma)$ si sa densité de probabilité est :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-m}{\sigma})^2}$$

Les lois de probabilités de la loi binomiale sont calculées grâce à la fonction `norm.pdf` de la bibliothèque `scipy.stats`.
Les lois de probabilité $\mathcal{N}(0, 1)$, $\mathcal{N}(2, 1.5)$ et $\mathcal{N}(2, 0.6)$ sont représentées figure 7. Les courbes ont une forme de Gaussienne, elles sont symétriques par rapport à la droite d'équation x = m et leur étalement augmente proportionnellement à σ.

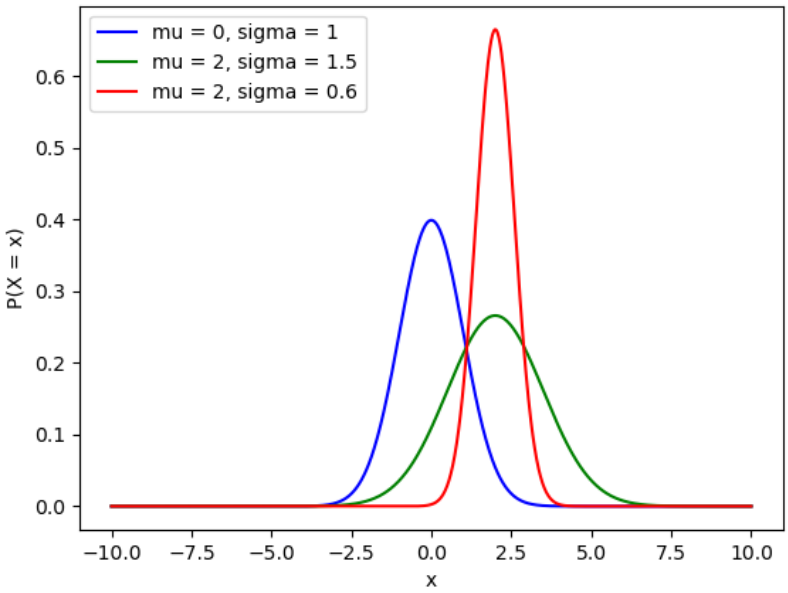

FIGURE 7 – Loi probabilité de la loi normale dans trois cas distincts

3.3 Simulation de données à partir d'une loi

Cas de la loi Normale

Nous allons générer quatre échantillons de taille 100, 1000, 10000 et 100000 selon la loi normale centrée réduite. Pour cela on utilise la fonction `random.normal` de la bibliothèque `numpy`. Ces échantillons sont représentés graphiquement sous forme d'histogramme et comparés à la courbe de la fonction de densité de la loi normale (voir figures 8, 9, 10 et 11).

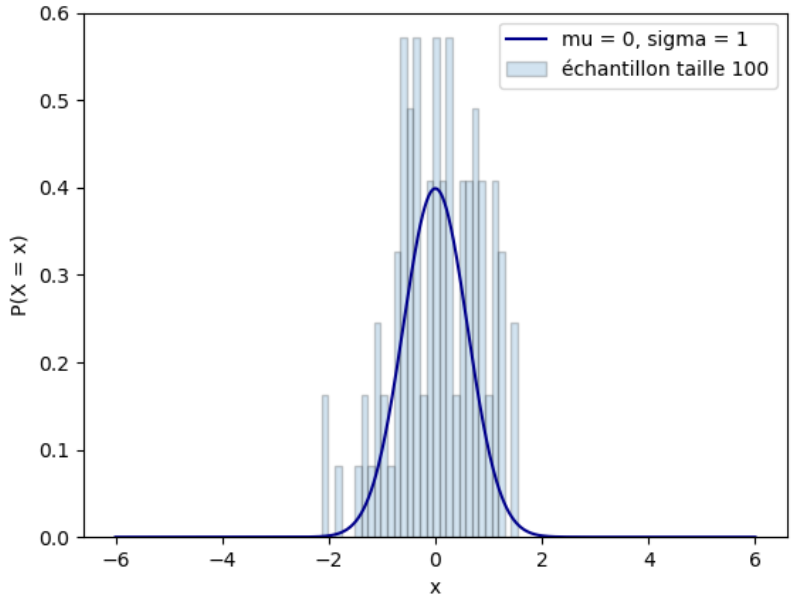

FIGURE 8 – Simulation de données à partir de la loi normale avec un échantillon de taille n=100

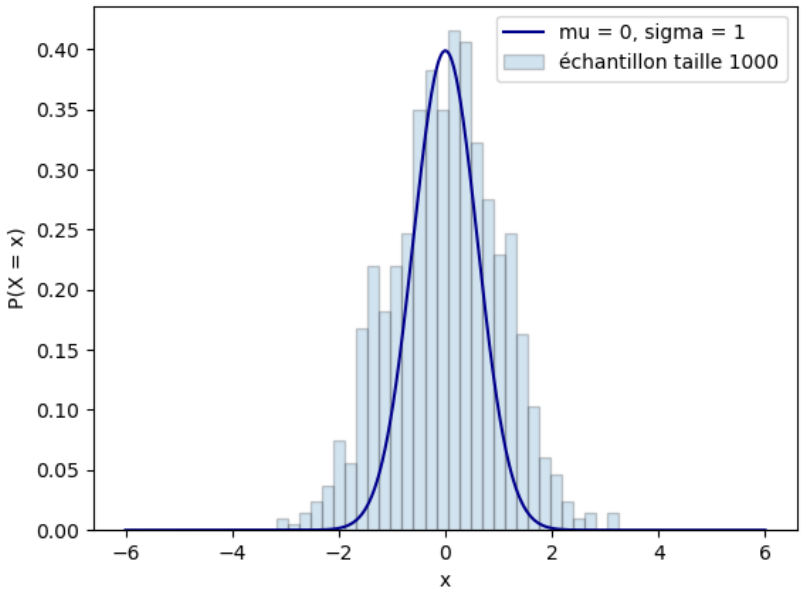

FIGURE 9 – Simulation de données à partir de la loi normale avec un échantillon de taille n=1000

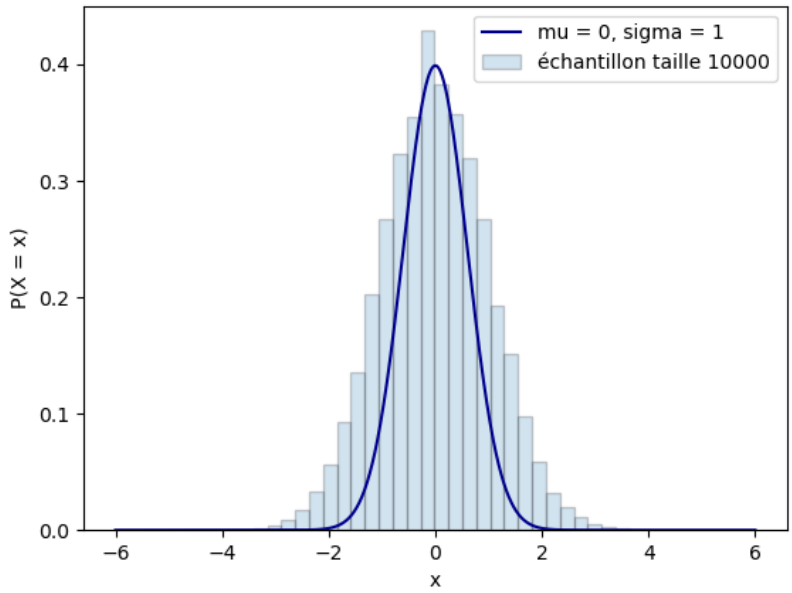

FIGURE 10 – Simulation de données à partir de la loi normale avec un échantillon de taille n=10 000

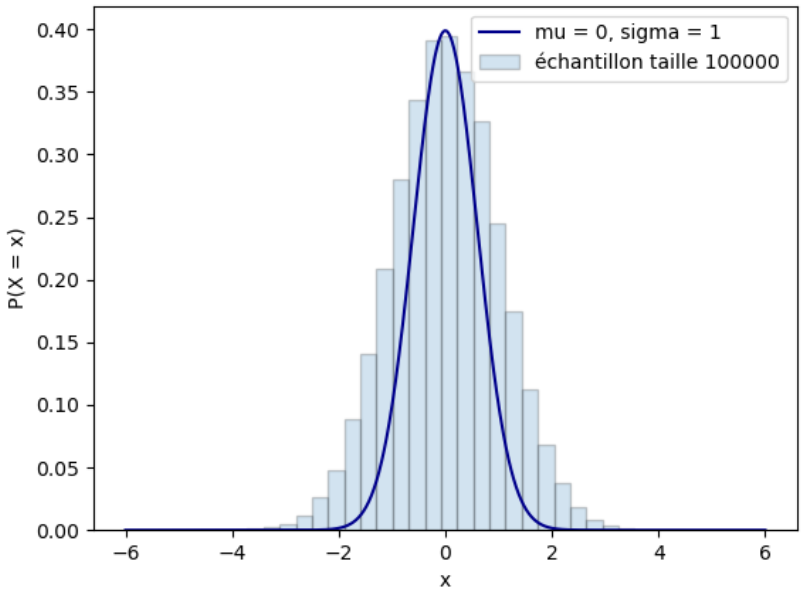

FIGURE 11 – Simulation de données à partir de la loi normale avec un échantillon de taille n=100 000

Les données obtenues prennent une forme de Gaussienne. Plus la taille de l'échantillon généré est grande et plus l'histogramme suit la courbe de densité de la loi normale.

3.4 Estimation de densité

Cas de la loi Normale

L'objectif est d'estimer la moyenne et l'écart-type pour des échantillons de diverses tailles et de les comparer avec la loi normale centrée réduite théorique $\mathcal{N}(0,1)$.

Les paramètres μ et σ sont estimés par maximum de vraisemblance grâce aux formules suivantes :

$$\text{Moyenne :} \quad \mu = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$\text{Variance :} \quad \sigma^2 = \frac{\sum_{i=1}^{n} (X_i - \mu)^2}{n-1}$$

$$\text{Écart-type :} \quad \sigma = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \mu)^2}{n-1}}$$

Trois échantillons de données de taille différente sont générés selon la loi normale centrée réduite $\mathcal{N}(0,1)$. Les moyennes et écart-type empiriques sont calculés dans le tableau 7 pour des échantillons de taille n=20, n=80 et n=150.

n	20	80	150
μ	0.1113	-0.0735	-0.0439
σ	1.0296	0.9272	0.9427

TABLE 7 – Estimation de la moyenne et de l'écart-type à partir d'échantillons de différentes tailles

Les moyennes et écart-type empiriques obtenus sont proches des valeurs théoriques μ=0 et σ=1. Les valeurs estimées se rapprochent globalement des valeurs théoriques lorsque n augmente.

Les courbes de densité théoriques et empiriques sont représentées figures 12, 13 et 14.

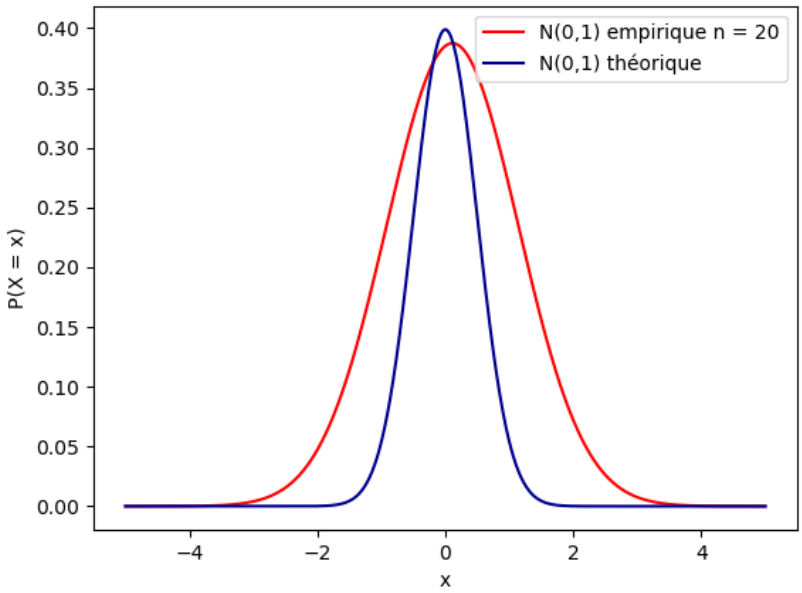

FIGURE 12 – Densité de la loi normale centrée réduite théorique (bleu) et densité empirique pour un échantillon de taille n = 20 (rouge)

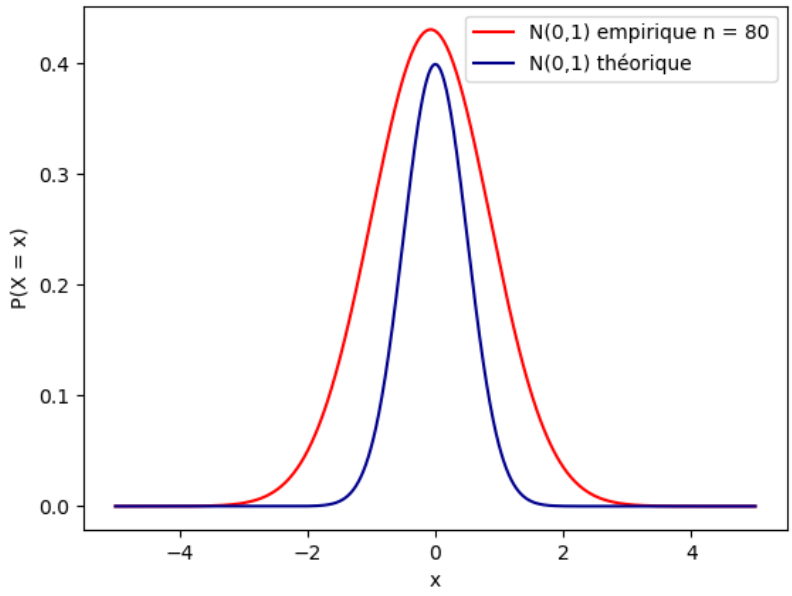

FIGURE 13 – Densité de la loi normale centrée réduite théorique (bleu) et densité empirique pour un échantillon de taille n = 80 (rouge)

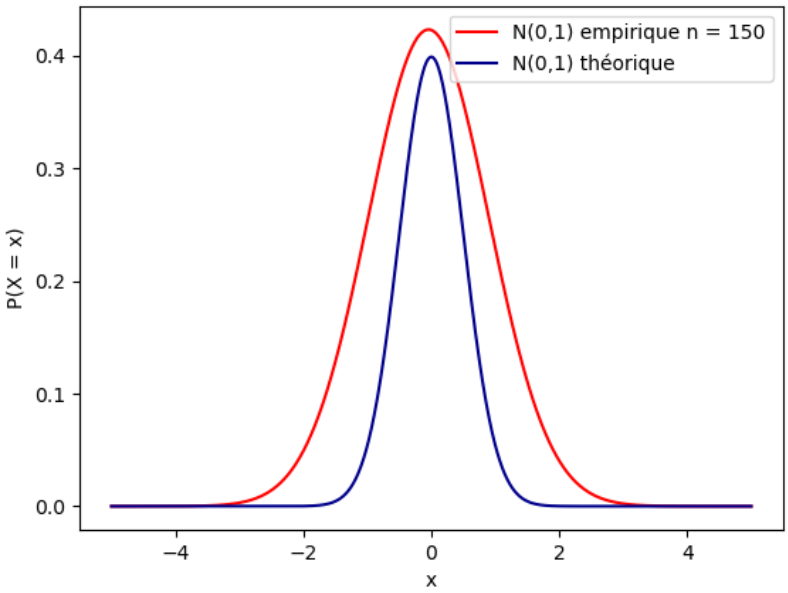

FIGURE 14 – Densité de la loi normale centrée réduite théorique (bleu) et densité empirique pour un échantillon de taille n = 150 (rouge)

On constate que les courbes obtenues ont bien une forme de Gaussienne et suivent globalement la courbe de la densité théorique même si elles ne se superposent pas, la courbe empirique étant plus étalée.

Cas de la loi exponentielle

Une variable aléatoire X suit une loi exponentielle de paramètre λ notée $\mathcal{E}(\lambda)$ telle que :

$$\text{Densité de probabilité : } f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{si x} \geq 0 \\ 0 & \text{si x} < 0 \end{cases}$$

$$\text{Fonction de répartition : } F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{si x} \geq 0 \\ 0 & \text{si x} < 0 \end{cases}$$

$$\text{Espérance : } E[X] = \frac{1}{\lambda}$$

$$\text{Variance : } var(X) = \frac{1}{\lambda^2}$$

La loi exponentielle permet de modéliser la durée de vie d'un phénomène sans vieillissement ou un temps d'attente. La variable aléatoire X représente alors la durée de vie d'un

phénomène.

Fonction de densité

Trois échantillons de données sont générés selon la loi exponentielle de paramètre $\lambda = 1$. Le paramètre λ estimé est calculé dans le tableau 8 pour des échantillons de taille n=20, n=80 et n=150. La fonction Python utilisée est `expon.pdf` (pour *probability density function*) de la bibliothèque `scipy.stats`.

n	20	80	150
λ	1.2132	1.0843	1.0297

TABLE 8 – Estimation du paramètre λ à partir d'échantillons de différentes tailles pour la densité de la loi exponentielle

Les courbes de densité théoriques et empiriques sont représentées figures 15, 16 et 17.

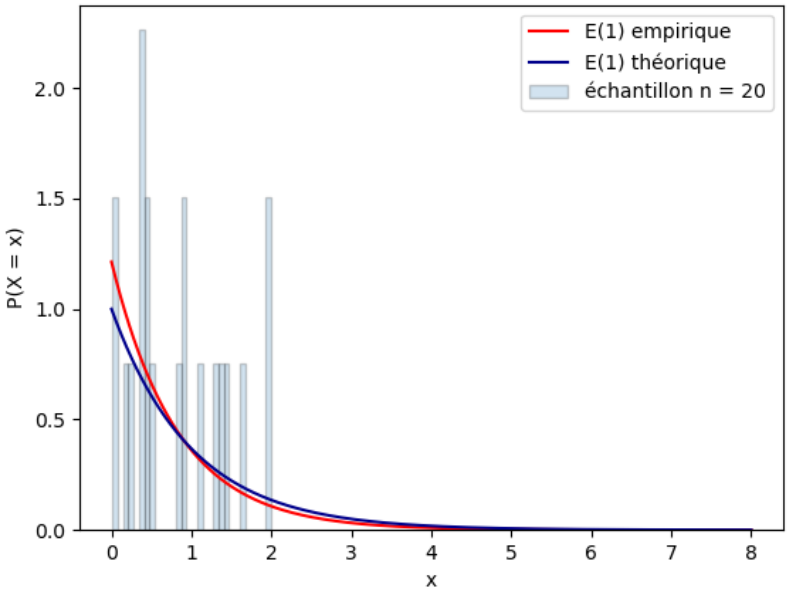

FIGURE 15 – Densité de la loi exponentielle théorique de paramètre $\lambda = 1$ (bleu) et densité empirique pour un échantillon de taille n = 20 (rouge)

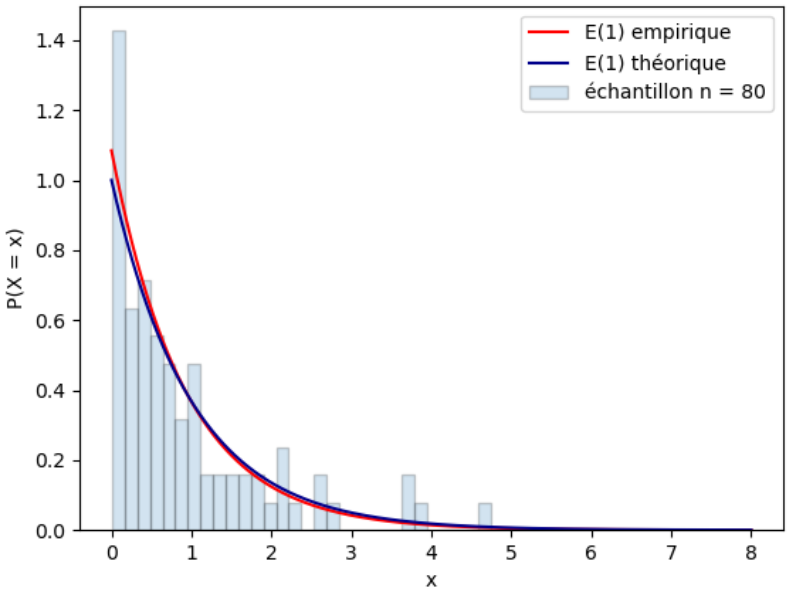

FIGURE 16 – Densité de la loi exponentielle théorique de paramètre $\lambda = 1$ (bleu) et densité empirique pour un échantillon de taille n = 80 (rouge)

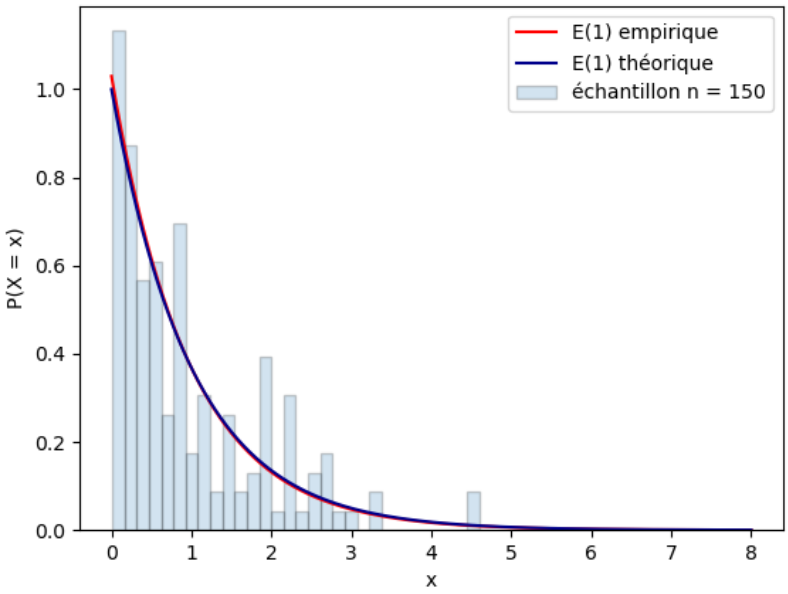

FIGURE 17 – Densité de la loi exponentielle théorique de paramètre $\lambda = 1$ (bleu) et densité empirique pour un échantillon de taille n = 150 (rouge)

Plus la taille de l'échantillon augmente et plus l'estimation est fiable. En effet la valeur du paramètre se rapproche de plus en plus de la valeur théorique $\lambda = 1$. La courbe empirique en rouge se rapproche de plus en plus de la courbe théorique en bleu à mesure que

n augmente. Les courbes théoriques et empiriques sont presque superposées pour n = 150.

Fonction de répartition

Le même travail est réalisé pour la fonction de répartition de la loi exponentielle. La fonction Python utilisée est `expon.cdf` (pour *cumulative density function*) de la bibliothèque `scipy.stats`. Les échantillons sont toujours de taille n=20, n=80 et n=150 et le paramètre théorique λ vaut 1. Le paramètre λ estimé est calculé dans le tableau 9 pour les trois échantillons.

n	20	80	150
λ	1.3273	0.9429	1.0380

TABLE 9 – Estimation du paramètre λ à partir d'échantillons de différentes tailles pour la fonction de répartition de la loi exponentielle

La valeur de λ empirique est de plus en plus proche de la valeur théorique $\lambda = 1$ à mesure que n augmente.

Les courbes de répartition théoriques et empiriques sont représentées figures 18, 19 et 20.

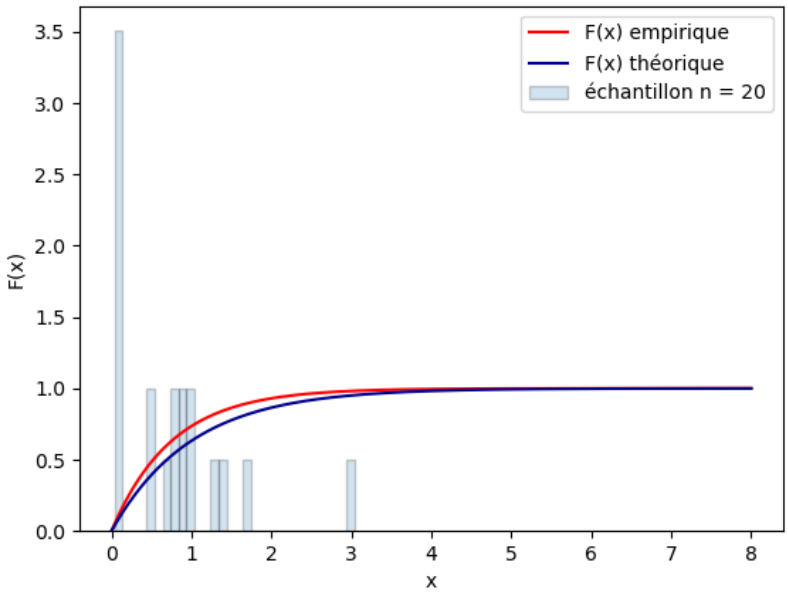

FIGURE 18 – Répartition de la loi exponentielle théorique de paramètre $\lambda = 1$ (bleu) et répartition empirique pour un échantillon de taille n = 20 (rouge)

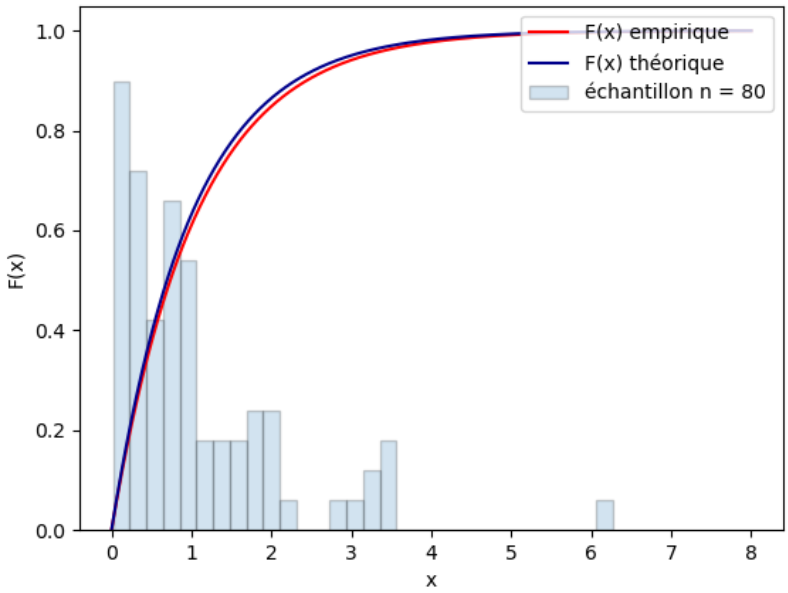

FIGURE 19 – Répartition de la loi exponentielle théorique de paramètre $\lambda = 1$ (bleu) et répartition empirique pour un échantillon de taille n = 80 (rouge)

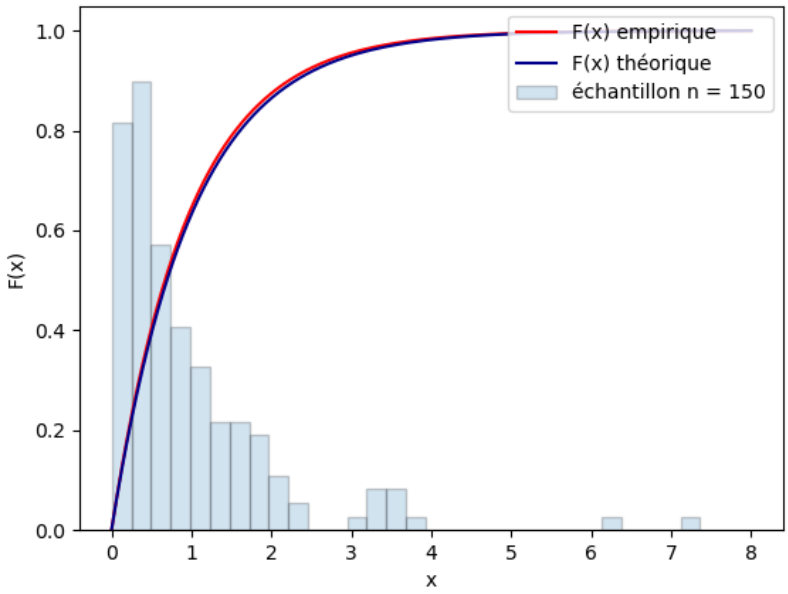

FIGURE 20 – Répartition de la loi exponentielle théorique de paramètre $\lambda = 1$(bleu) et répartition empirique pour un échantillon de taille n = 150 (rouge)

Plus la taille de l'échantillon n augmente et plus les courbes empiriques et théoriques sont proches.

Variation du paramètre λ

La densité de la loi exponentielle est tracée pour quatre valeurs de λ différentes afin de voir l'influence du paramètre λ sur la courbe obtenue. Les lois $\mathcal{E}(0.5)$, $\mathcal{E}(1)$, $\mathcal{E}(1.5)$ et $\mathcal{E}(2)$ sont tracées figure 21.

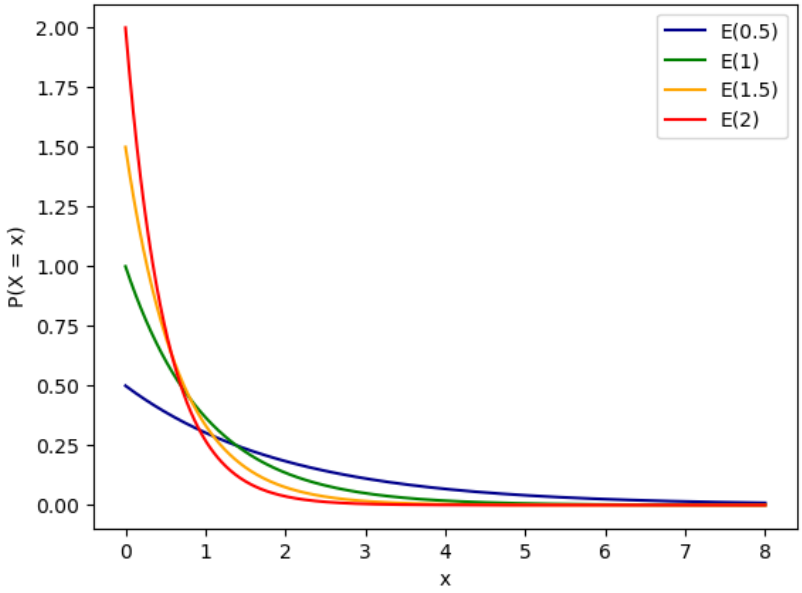

FIGURE 21 – Densité de la loi exponentielle pour différentes valeurs de paramètre λ

Plus la valeur du paramètre λ augmente et plus la fonction de densité est grande pour des petites valeurs de x et plus elle décroît rapidement lorsque x augmente.

4 Intervalles de confiance

L'objectif est de déterminer les intervalles de confiance à différents seuils de confiance sur trois échantillons de données.

Soit X une variable aléatoire suivant un loi normale $\mathcal{N}(\mu, \sigma^2)$, et $X_1, X_2, ..., X_n$ n variables selon la loi X.

$$\text{Moyenne empirique :} \quad \bar{X}_n = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$\text{Variance empirique :} \quad S_n^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X}_n)^2}{n}$$

$$\text{Intervalle de confiance avec variance connue : } IC(\mu) = [\bar{X}_n - u\frac{\sigma}{\sqrt{n}}, \bar{X}_n + u\frac{\sigma}{\sqrt{n}}]$$

$$\text{où u est le fractile d'ordre } 1 - \frac{\alpha}{2} \text{ de la loi } \mathcal{N}(0,1)$$

$$\text{Intervalle de confiance avec variance inconnue : } IC(\mu) = [\bar{X}_n - t\frac{S_n}{\sqrt{n}}, \bar{X}_n + t\frac{S_n}{\sqrt{n}}]$$

$$\text{où t est le fractile d'ordre } 1 - \frac{\alpha}{2} \text{ de la loi de Student } St(n-1)$$

Le programme `tp3.py` contient les fonctions suivantes :

- `moyenne` prend en entrée un liste et calcule sa moyenne empirique.
- `variance` prend en entrée une liste et calcule sa variance empirique.
- Lorsque la variance est inconnue : `intervalle_confiance` prend en entrée une liste et le niveau de confiance, calcule la moyenne empirique, le fractile grâce à la fonction `scipy.stats.t.ppf`, l'écart-type grâce à la variance empirique et retourne l'intervalle de confiance.
- Lorsque la variance est connue : `intervalle_confiance2` prend en entrée une liste, le niveau de confiance et la variance, calcule la moyenne empirique, le fractile grâce à la fonction `scipy.stats.norm.ppf`, l'écart-type grâce à la variance donnée en argument et retourne l'intervalle de confiance.

Les fonctions utilisées pour les fractiles découlent de la bibliothèque `scipy.stats` et retournent la valeur voulue avec en entrée l'ordre $1 - \frac{\alpha}{2}$ demandé. Si l'intervalle est demandé au niveau 95%, alors $\alpha = 0.05$. Si l'intervalle est demandé au niveau 99%, alors $\alpha = 0.01$.

4.1 Problème 1

On considère un échantillon de 16 pots de confiture d'une certaine marque, on mesure le poids en kg de chaque pot dans le tableau 10.

Poids/kg	0.499	0.509	0.501	0.494	0.498	0.497	0.504	0.506
0.502	0.496	0.495	0.493	0.507	0.505	0.503	0.491	

TABLE 10 – Poids des pots de confiture

Le poids en kg d'un pot de confiture peut être décrit par une variable aléatoire suivant une loi normale $\mathcal{N}(\mu, \sigma^2)$.

La liste des poids des pots de confiture est représentée par la variable `p_confiture` et sert en premier lieu à calculer la moyenne empirique. On affiche ensuite l'histogramme de fréquence (voir figure 22) avec la bibliothèque `matplotlib.pyplot` et plus particulièrement les fonctions `hist` et `show`. Pour ce premier problème, la variance est inconnue. On utilise donc la fonction `intervalle_confiance` avec cette même liste et un niveau de 0.05 pour 95%, puis 0.01 pour 99%.

La moyenne empirique obtenue est $\bar{X}_{16} = 0.5$ kg.
Les intervalles de confiance obtenus sont les suivants :
- pour un seuil de 95% : IC = [0.4972, 0.5028] kg
- pour un seuil de 99% : IC = [0.4961, 0.5039] kg

Les intervalles sont quasiment similaires. On a 95% de chance pour qu'un pot de confiture pris au hasard ait un poids compris entre 0.4972 et 0.5028 kg. On a 99% de chance pour qu'un pot de confiture pris au hasard ait un poids compris entre 0.4961 et 0.5039 kg.

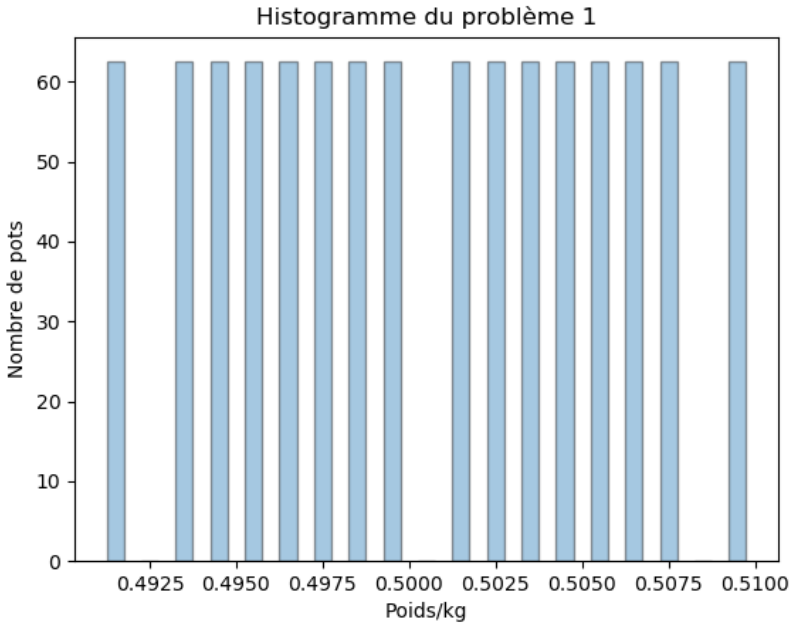

FIGURE 22 – Histogrammes des fréquences du poids des pots de confiture

On se donne un deuxième jeu de données dans le tableau 11 décrivant la masse de l'avocat provenant d'une ferme au Mexique.

Poids/g	85.06	91.44	87.93	89.02	87.28	82.34	86.23	84.16
88.56	90.45	84.91	89.90	85.52	86.75	88.54	87.90	

TABLE 11 – Poids des avocats

La liste des poids d'avocats est représentée par la variable `p_avocat`, et comme dans le premier cas, la variance est inconnue. On utilise donc la fonction `intervalle_confiance` avec `p_avocat` et un niveau de 0.05 pour 95%.
L'intervalle de confiance obtenu est le suivant : IC = [85.9826, 88.5162] g pour un seuil de 95%.

4.2 Problème 2

Une compagnie aérienne souhaite étudier le pourcentage de voyageurs satisfaits par ses services, on en a interrogé 500 choisis au hasard. Parmi eux, 95 se disent satisfaits. Déterminer un intervalle de confiance pour la proportion inconnue de voyageurs satisfaits au niveau 99%.

La population est représentée par la variable `satisfaction`, dont les 95 premiers termes sont à 1 (clients satisfaits) et la suite est à 0 (clients insatisfaits). Ici, la variance est inconnue. On utilise donc la fonction `intervalle_confiance` avec comme arguments `satisfaction` et un niveau de 0.01 pour un seuil de confiance de 99%.
L'intervalle de confiance obtenu est le suivant : IC = [0.1465, 0.2375] pour un seuil de 99%.

4.3 Problème 3

Simuler un échantillon de taille n de $\mathcal{B}(p = \frac{1}{2})$ indépendantes. Calculer numériquement un intervalle de confiance du paramètre, de niveau de confiance de 95%.

Pour ce problème, la taille de l'échantillon est saisie au clavier par l'utilisateur et on fait appel à la fonction `bernoulli.rvs` de la bibliothèque `scipy.stats` qui nous donne selon un paramètre p et la taille de l'échantillon n, une liste de résultat de n épreuves de Bernoulli. Ici, la variance est connue, car dans la cas d'une loi de Bernoulli, on a $var(X) = p(1-p)$. On simule un échantillon de taille n selon une loi de Bernoulli $\mathcal{B}(\frac{1}{2})$. On appelle donc la fonction `intervalle_confiance2` avec comme arguments `echantillon`, `var` et 0.05 pour un seuil de confiance de 95%. Plusieurs intervalles de confiance au seuil de 95% sont calculés et présentés dans le tableau 12.

n	IC
10	[0.1901, 0.8099]
50	[0.3414, 0.6186]
100	[0.3620, 0.5580]
500	[0.4842, 0.5718]
1000	[0.4740, 0.5360]
5000	[0.4725, 0.5003]
10000	[0.4894, 0.5090]
100000	[0.4946, 0.5008]

TABLE 12 – Intervalles de confiance au seuil de 95% pour le problème 3

On constate que plus la taille de l'échantillon augmente et plus l'intervalle de confiance est restreint.

5 Conclusion

Ce projet a permis d'appliquer les concepts et lois étudiés en cours. Les méthodes de régression linéaires des moindres carrés, matricielle et de descente de gradient ont pu être appliquées et comparées. Les fonctions de densité de probabilité théoriques et empiriques des lois binomiales, normales et exponentielles ont pu être tracées et comparés. Enfin, les intervalles de confiance au seuil de 95% et 99% ont été calculés sur plusieurs jeux de données.