Projet mathématique

ALBERTINI PRAVEDNYI

4 juin 2023

1 Développement mathématiques

1.1 Question 1.a

Dans cette premièer question, nous allons montrer que les lignes d'isodensité de f_Z sont des ellipses.

Par définition, une ligne d'isodensité est un ensemble de point du plan pour lesquels la densité de probabilité est constante.

Si l'on prend une constante K, on a donc :

$$f_Z(z) = K \tag{1}$$

En substituant $f_Z(z)$ par son expression, on obtient :

$$K = \frac{1}{\sqrt{(2\pi)^2 det\Sigma}} exp(-\frac{1}{2}(z-\mu)^t \Sigma^{-1}(z-\mu)) \tag{2}$$

Pour notre étude, nous nous limiterons au cas où la matrice de covariance Σ est définie positive. Nous pouvons donc utiliser la décomposition spectrale de Σ.

Cette propriété nous apprend que la matrice, étant symétrique et définie positive, est diagonalisable à l'aide d'une matrice de passage orthogonale, et que ses valeurs propres sont positives.

$$\Sigma = U\Lambda U^T \tag{3}$$

Remarquons que Λ est une matrice diagonale composée de λ_1 et λ_2, les valeurs propres de Σ. Le déterminant de Σ vaut donc $\lambda_1\lambda_2$.

En conséquence, Σ^{-1} est la matrice diagonale composée des termes $\frac{1}{\lambda_1}$ et $\frac{1}{\lambda_2}$.

Nous pouvons donc écrire

$$\Sigma^{-1} = U\Lambda^{-1}U^T \tag{4}$$

En substituant (4) dans (2) et en remplacant det(Σ), on obtient :

$$K = \frac{1}{\sqrt{(2\pi)^2 \lambda_1\lambda_2}} exp(-\frac{1}{2}(z-\mu)^t\ U\Lambda^{-1}U(z-\mu)) \tag{5}$$

On souhaite poser $a =^t P(z-\mu)$. On a donc la multiplication suivante :

$$a = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x-\mu_1 \\ y-\mu_2 \end{bmatrix} \tag{6}$$

On trouve :

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} cos\theta(x-\mu_1) - sin\theta(y-\mu_2) \\ sin\theta(x-\mu_1) + cos\theta(y-\mu_2) \end{bmatrix} \tag{7}$$

On veut introduire a dans (5). Puisque $a =^t P(z-\mu)$, on a $(z-\mu) = Pa$. On remplace :

$$K = \frac{1}{\sqrt{(2\pi)^2 \lambda_1\lambda_2}} exp(-\frac{1}{2}(Ua)^t\ U\Lambda^{-1t}UUa) \tag{8}$$

Puisque $(Ua)^t = a^t U^t$, on a deux fois le produit $U^t U$.
Or, U étant une matrice orthogonale, on a $U^t U = I_n$. Ainsi, (8) se simplifie en

$$K = \frac{1}{\sqrt{(2\pi)^2 \lambda_1 \lambda_2}} exp(-\frac{1}{2} a^t \Lambda^{-1} a) \tag{9}$$

En calculant le produit $a^t \Lambda^{-1} a$, on trouve :

$$a^t \Lambda^{-1} a = \begin{bmatrix} a1 & a2 \end{bmatrix} * \begin{bmatrix} \frac{1}{\lambda_1} & 0 \\ 0 & \frac{1}{\lambda_2} \end{bmatrix} * \begin{bmatrix} a1 \\ a2 \end{bmatrix} = \frac{a_1^2}{\lambda_1} + \frac{a_2^2}{\lambda_2} \tag{10}$$

En remplaçant (10) dans (9), on obtient :

$$K = \frac{1}{\sqrt{(2\pi)^2 \lambda_1 \lambda_2}} exp(-\frac{1}{2}(\frac{a_1^2}{\lambda_1} + \frac{a_2^2}{\lambda_2})) \tag{11}$$

En passant la constante de l'autre coté et en appliquant un logarithme aux deux membres, on obtient :

$$-2 * ln(K 2\pi \sqrt{\lambda_1 \lambda_2}) = \frac{a_1^2}{\lambda_1} + \frac{a_2^2}{\lambda_2} \tag{12}$$

En divisant tout par le membre de gauche, on obtient :

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{(cos\theta(x-\mu_1) + sin\theta(y-\mu_2))^2}{\lambda_1 * -2 * ln(K 2\pi \sqrt{\lambda_1 \lambda_2})} + \frac{(-sin\theta(x-\mu_1) + cos\theta(y-\mu_2))^2}{\lambda_2 * -2 * ln(K 2\pi \sqrt{\lambda_1 \lambda_2})} \tag{13}$$

On reconnait clairement l'équation cartésienne d'une ellipse. θ est l'angle de l'ellipse par rapport à l'horizontale, et le centre de l'ellipse sera donné par le vecteur μ. Les demi axes vaudront respectivement :

$$a = \sqrt{\lambda_1 * -2 * ln(K 2\pi \sqrt{\lambda_1 \lambda_2})} \tag{14}$$

$$b = \sqrt{\lambda_2 * -2 * ln(K 2\pi \sqrt{\lambda_1 \lambda_2})} \tag{15}$$

1.2 Question 1.b

Nous cherchons désormais à calculer la probabilité qu'un point tiré selon la loi de Z appartiennent à la surface interne S_K, surface délimitée par l'ellipse d'isodensité K. Ensuite, nous ferons l'inverse : pour une probabilité donnée, on cherchera l'ellipse d'isodensité K qui vérifie $P(Z \in S_K) = p$.

1.2.1 Calcul de la probabilité pour une ellipse donnée

Pour déterminer la probabilité qu'un point tiré selon la loi de Z appartienne à une ellipse donnée, il faut calculer l'intégrale de $f_Z(z)$ sur la surface de l'ellipse :

$$P(Z \in S_K) = \iint_{S_K} f_Z(x,y)\, dx\, dy \tag{16}$$

On remplace $f_Z(x,y)$ par l'expression simplifié que nous avions trouvée en (11).

$$P(Z \in S_K) = \iint_{S_K} \frac{1}{\sqrt{(2\pi)^2 \lambda_1 \lambda_2}} exp(-\frac{1}{2}(\frac{a_1^2}{\lambda_1} + \frac{a_2^2}{\lambda_2}))\, dx\, dy \tag{17}$$

En substituant encore a_1 et a_2 par leurs valeurs exprimées en (7), on obtient :

$$P(Z \in S_K) = \iint_{S_K} \frac{1}{\sqrt{(2\pi)^2 \lambda_1 \lambda_2}} \exp\left(-\frac{1}{2}\left(\frac{(cos\theta(x-\mu_1) - sin\theta(y-\mu_2))^2}{\lambda_1} + \frac{(sin\theta(x-\mu_1) + cos\theta(y-\mu_2))^2}{\lambda_2}\right)\right) dx\, dy \tag{18}$$

Pour résoudre cette belle intégrale, nous allons faire un premier changement de variable. On pose :

$$x' = \frac{cos(\theta)(x-\mu_1) - sin(\theta)(y - mu_2)}{\sqrt{\lambda_1}} \tag{19}$$

$$y' = \frac{sin(\theta)(x-\mu_1) + cos(\theta)(y - mu_2)}{\sqrt{\lambda_2}} \tag{20}$$

Pour appliquer ce changement de variable à une intégrale double, nous allons utiliser le théorème de changement de variable pour une intégrale double. Or, pour utiliser ce théorème, il faut d'abord montrer que l'application linéaire du changement de variable est bijective.
Pour cela, nous allons passer par la matrice jacobienne. En effet, si le déterminant de la matrice jacobienne n'est pas nul, le changement de variable associé est bijectif. Pour obtenir la matrice jacobienne, on calcule :

$$J = \begin{bmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{bmatrix} \tag{21}$$

Si l'on développe x' et y', on voit assez facilement qu'il y a un terme dépendant de x, un terme dépendant de y, et deux termes constants.

$$x' = \frac{cos(\theta)x}{\sqrt{\lambda_1}} - \frac{cos(\theta)\mu_1}{\sqrt{\lambda_1}} - \frac{sin(\theta)y}{\sqrt{\lambda_1}} + \frac{sin(\theta)\mu_2}{\sqrt{\lambda_1}} \tag{22}$$

$$y' = \frac{sin(\theta)x}{\sqrt{\lambda_2}} - \frac{sin(\theta)\mu_1}{\sqrt{\lambda_2}} + \frac{cos(\theta)y}{\sqrt{\lambda_2}} - \frac{cos(\theta)\mu_2}{\sqrt{\lambda_2}} \tag{23}$$

On isole x et y :

$$x = x'\frac{\sqrt{\lambda_1}}{cos(\theta)} + \mu_1 + tan(\theta)y - tan(\theta)\mu_2 \tag{24}$$

$$y = -tan(\theta)x + tan(\theta)\mu_1 + y'\frac{\sqrt{\lambda_2}}{cos(\theta)} + \mu_2 \tag{25}$$

On remplace y par son expression dans l'expression de x :

$$\begin{aligned} x &= x'\frac{\sqrt{\lambda_1}}{\cos(\theta)} + \mu_1 + \tan(\theta)(-\tan(\theta)x + \tan(\theta)\mu_1 + y'\frac{\sqrt{\lambda_2}}{\cos(\theta)} + \mu_2) - \tan(\theta)\mu_2 \\ \Leftrightarrow x(1+tan^2(\theta)) &= x'\frac{\sqrt{\lambda_1}}{\cos(\theta)} - y'tan(\theta)\frac{\sqrt{\lambda_2}}{cos(\theta)} + \mu_1 - \tan(\theta)(\tan(\theta)\mu_1 + \mu_2) - \tan(\theta)\mu_2 \\ \Leftrightarrow x &= \frac{x'\frac{\sqrt{\lambda_1}}{\cos(\theta)} - y'tan(\theta)\frac{\sqrt{\lambda_2}}{cos(\theta)} + \mu_1 - \tan(\theta)(\tan(\theta)\mu_1 + \mu_2) - \tan(\theta)\mu_2}{(1+tan^2(\theta))} \end{aligned} \tag{26}$$

De même, on remplace x par son expression dans l'expression de y :

$$\begin{aligned} y &= -tan(\theta)(x'\frac{\sqrt{\lambda_1}}{cos(\theta)} + \mu_1 + tan(\theta)y - tan(\theta)\mu_2) + tan(\theta)\mu_1 + y'\frac{\sqrt{\lambda_2}}{cos(\theta)} + \mu_2 \\ \Leftrightarrow y(1+tan^2(\theta)) &= -tan(\theta)(x'\frac{\sqrt{\lambda_1}}{cos(\theta)} + \mu_1 - tan(\theta)\mu_2) + tan(\theta)\mu_1 + y'\frac{\sqrt{\lambda_2}}{cos(\theta)} + \mu_2 \\ \Leftrightarrow y &= \frac{-tan(\theta)x'\frac{\sqrt{\lambda_1}}{cos(\theta)} + y'\frac{\sqrt{\lambda_2}}{cos(\theta)} - tan(\theta)(\mu_1 - tan(\theta)\mu_2) + tan(\theta)\mu_1 + \mu_2}{(1+tan^2(\theta))} \end{aligned} \tag{27}$$

Ainsi, on a deux expressions de la forme ax + by + c. La dérivée partielle selon x vaut a, et la dérivée partielle selon y vaut b. On trouve alors la matrice jacobienne suivante :

$$J = \begin{bmatrix} \frac{\sqrt{\lambda_1}}{\cos(\theta)}(1+tan^2(\theta)) & \frac{tan(\theta)\sqrt{\lambda_2}}{cos(\theta)}(1+tan^2(\theta)) \\ \frac{-tan(\theta)\sqrt{\lambda_1}}{cos(\theta)}(1+tan^2(\theta)) & \frac{\sqrt{\lambda_2}}{cos(\theta)}(1+tan^2(\theta)) \end{bmatrix} \tag{28}$$

On calcule le déterminant de J. On a :

$$|J| = \sqrt{\lambda_1 \lambda_2} \neq 0 \tag{29}$$

Le déterminant n'étant pas nul, l'application linéaire du changement de variable est bijective. On peut alors appliquer le théorème de changement de variable pour une intégrale double. Ainsi, la nouvelle intégrale définie sur la surface S' vaut :

$$P(Z \in S') = \frac{1}{2\pi\sqrt{\lambda_1 \lambda_2}} \iint_{S_K} exp\left(-\frac{1}{2}(x'^2 + y'2)\right) \cdot \sqrt{\lambda_1 \lambda_2}\, dx'\, dy' \tag{30}$$

Pour encore simplifier le calcul de l'intégrale, nous allons effectuer un second changement de variable. Pour cela, nous allons passer aux coordonnées polaires. Puisque l'on travaille sur une ellipse, ces changements de variables sont :

$$x' = rcos(\theta) \tag{31}$$

$$y' = rsin(\theta) \tag{32}$$

Encore une fois, il est nécessaire de déterminer la matrice jacobienne de cette transformation. Pour cela, il faut déterminer les quatres dérivées partielles :

$$J = \begin{bmatrix} \frac{\partial x'}{\partial r} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial r} & \frac{\partial y'}{\partial \theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -rsin(\theta) \\ sin(\theta) & rcos(\theta \end{bmatrix}$$

Calculons maintenant le déterminant de cette matrice :

$$\begin{aligned} &= r \cdot cos(\theta)^2 - (-r \cdot sin(\theta)^2) \\ &= r \cdot (cos(\theta)^2 + sin(\theta)^2) \\ &= r \end{aligned} \tag{33}$$

Ainsi, le jacobien de transformation pour le passage aux coordonnées polaires est égal à r. Le jacobien n'étant pas nul, on peut appliquer le théorème de changement de variable :

$$P(Z \in S') = \frac{1}{2\pi} \int_0^{2\pi} \int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr\, d\theta \tag{34}$$

Ici, le contenu de notre intégrale ne dépend pas de θ. Ainsi, on peut considérer $\int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr$ comme une constante. L'intégrale d'une constante étant la constante multipliée par la variable puissance 1, on a :

$$\begin{aligned} P(Z \in S') &= \frac{1}{2\pi} \left(\left[\int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr \right) \cdot \theta \right]_0^{2\pi} \\ &= \frac{1}{2\pi} \left(\int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr \right) \cdot [\theta]_0^{2\pi} \\ &= \frac{1}{2\pi} \left(\int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr \right) \cdot (2\pi - 0) \\ &= \int_0^r exp\left(-\frac{r^2}{2}\right) r\, dr \end{aligned} \tag{35}$$

Il nous reste alors à intégrer selon r. Pour cela, nous allons faire un troisième et dernier changement de variable : $u = \frac{r^2}{2}$ et $du = r \cdot dr$. On peut alors calculer l'intégrale aisément :

$$\begin{aligned}\int_0^r exp\left(-\frac{r^2}{2}\right) r\,dr &= \int_0^u exp(-u)\,du \\ &= [exp(-u)]_0^u \\ &= -exp(-u) - (-exp(-0)) \\ &= -e^{-u} + 1 \end{aligned} \tag{36}$$

Maintenant, il convient de remonter les changements de variable.
En remplaçant u par $\frac{r^2}{2}$, on trouve le résultat suivant :

$$P(Z \in S') = -exp\left(-\frac{r^2}{2}\right) + 1 \tag{37}$$

On souhaite maintenant remplacer r par x' et y'. Pour cela, nous allons partir de (27) et (28). On élève les deux équations au carré :

$$(x')^2 = r^2 cos^2(\theta) \tag{38}$$

$$(y')^2 = r^2 sin^2(\theta) \tag{39}$$

On additionne les équations, et on cherche à isoler r tout en supprimant θ :

$$\begin{aligned}(x')^2 + (y')^2 &= r^2 cos^2(\theta) + r^2 sin^2(\theta) \\ \Leftrightarrow (x')^2 + (y')^2 &= r^2(cos^2(\theta) + sin^2(\theta)) \\ \Leftrightarrow r^2 &= (x')^2 + (y')^2 \\ \Leftrightarrow r &= \sqrt{(x')^2 + (y')^2} \end{aligned} \tag{40}$$

Ainsi, on a pu exprimer r en fonction de x' et y'. On remplace r^2 dans l'intégrale :

$$P(Z \in S') = -exp\left(-\frac{(x')^2 + (y')^2}{2}\right) + 1 \tag{41}$$

Enfin, en remplaçant x' et y' par leurs valeurs en (19) et en (20), on trouve l'expression finale :

$$\begin{aligned}P(Z \in S') &= exp\left(-\frac{(x')^2 + (y')^2}{2}\right) + 1 \\ &= -exp\left(-\frac{1}{2}\left(\left(\frac{cos(\theta)(x-\mu_1) + sin(\theta)(y-\mu_2)}{\sqrt{\lambda_1}}\right)^2 + \left(\frac{-sin(\theta)(x-\mu_1) + cos(\theta)(y-\mu_2)}{\sqrt{\lambda_2}}\right)^2\right)\right) + 1\end{aligned}$$

Or, d'après (11) et (7) on a également :

$$\begin{aligned}K &= \frac{1}{2\pi\sqrt{\lambda_1\lambda_2}} exp(-\frac{1}{2}(\frac{a_1^2}{\lambda_1} + \frac{a_2^2}{\lambda_2})) \\ \Leftrightarrow -2\pi K\sqrt{\lambda_1\lambda_2} &= -exp\left(-\frac{1}{2}\left(\frac{(cos\theta(x-\mu_1) + sin\theta(y-mu_2)}{\sqrt{\lambda_1}}\right)^2 + \left(\frac{(-sin\theta(x-\mu_1) + cos\theta(y-mu_2)}{\sqrt{\lambda_2}}\right)^2\right)\end{aligned} \tag{42}$$

Ainsi, on peut lier K et $P(Z \in S')$:

$$P(Z \in S') = 1 - 2\pi K\sqrt{\lambda_1\lambda_2} \tag{43}$$

1.2.2 Calcul de l'ellipse d'isodensité K pour une probabilité p donnée

Pour trouver l'ellipse d'isodensité K correspondant à une probabilité p donnée, il suffit d'isoler K dans (44)

$$\begin{aligned} P(Z \in S') &= 1 - 2\pi K\sqrt{\lambda_1\lambda_2} \\ \Leftrightarrow K &= \frac{1-p}{2\pi\sqrt{\lambda_1\lambda_2}} \end{aligned} \tag{44}$$

1.3 Question 2

Nous allons ici chercher à déterminer un estimateur de μ et un estimateur de Σ.
Pour cela, nous allons utiliser la méthode du maximum de vraisemblance.

1.3.1 Fonction de (log)vraisemblance

Pour déterminer la fonction de vraisemblance, il suffit de prendre le produit de la fonction de densité évaluée en chaque point :

$$L(\mu, \Sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}} \exp\left(-\frac{1}{2}(z_i - \mu)^T \Sigma^{-1}(z_i - \mu)\right) \tag{45}$$

Pour étudier le maximum de $L(\mu, \Sigma)$, nous allons en réalité étudier la fonction de log-vraisemblance.

$$\log L(\mu, \Sigma) = \sum_{i=1}^{n} \left(\log\left(\frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}}\right) - \frac{1}{2}(z_i - \mu)^T \Sigma^{-1}(z_i - \mu)\right) \tag{46}$$

1.3.2 Estimateur de μ

Pour obtenir l'estimateur de μ, il faut trouver le maximum de la fonction de log-vraisemblance. Cela revient à trouver la valeur de μ telle que la dérivée partielle de $\log L(\mu, \Sigma)$ par μ vaille 0.
Pour simplifier le calcul de la dérivée, on sépare la fonction de log-vraisemblance en deux :

$$\log L(\mu, \Sigma) = \sum_{i=1}^{n} \left(\log\left(\frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}}\right)\right) - \frac{1}{2}\sum_{i=1}^{n}\left((z_i - \mu)^T \Sigma^{-1}(z_i - \mu)\right) \tag{47}$$

Ici, la première somme ne dépendant pas de μ, sa dérivée partielle sera nulle. Pour la deuxième somme, la dérivée partielle vaut :

$$-\frac{1}{2}\left(-2\Sigma^{-1}\sum_{i=1}^{n}(z_i - \mu)\right) = \Sigma^{-1}\sum_{i=1}^{n}(z_i - \mu) \tag{48}$$

On cherche donc à résoudre cette équation :

$$\frac{\partial}{\partial\mu}\log L(\mu, \Sigma) = \Sigma^{-1}\sum_{i=1}^{n}(z_i - \mu) = 0 \tag{49}$$

En multipliant les deux membres à gauche par Σ et en isolant μ, on obtient :

$$\sum_{i=1}^{n}(z_i) - \sum_{i=1}^{n}(\mu) = 0 \Leftrightarrow \sum_{i=1}^{n}(z_i) = \sum_{i=1}^{n}(\mu) \Leftrightarrow \sum_{i=1}^{n} z_i = n\mu \tag{50}$$

Ainsi, l'estimateur de μ, que l'on nomme $\hat{\mu}$, vaut :

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{51}$$

On remarque que l'estimateur obtenu correspond à la moyenne des données. Ce résultat parait plutôt normal, puisque les points sont équitablement répartis autour du centre de l'ellipse.

1.3.3 Estimateur de Σ

Pour obtenir l'estimateur de Σ, il est nécessaire de dériver $\log L(\mu, \Sigma)$ par Σ. Pour rappel, la fonction de log-vraisemblance vaut :

$$\log L(\mu, \Sigma) = \sum_{i=1}^{n} \left(\log \left(\frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}} \right) - \frac{1}{2}(z_i - \mu)^T \Sigma^{-1} (z_i - \mu) \right) \tag{52}$$

Malheureusement, Σ est une matrice, ce qui suppose une dérivation matricielle longue et complexe. Pour simplifier le calcul, nous allons diviser la somme en deux parties. Pour la première, nous avons :

$$\begin{aligned} \sum_{i=1}^{n} \left(\log \left(\frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}} \right) \right) &= \sum_{i=1}^{n} \left(-\frac{1}{2} \log((2\pi)^2 \det(\Sigma) \right) \\ &= \sum_{i=1}^{n} \left(-\frac{1}{2} \log((2\pi)^2) + \log(\det(\Sigma)) \right) \\ &= -\frac{n}{2} \log(4\pi^2) + -\frac{n}{2} \log(\det(\Sigma)) \end{aligned} \tag{53}$$

Pour simplifier la seconde partie, nous allons passer par la trace. En effet, on a :

$$\sum_{i=1}^{n} \left(-\frac{1}{2}(z_i - \mu)^T \Sigma^{-1} (z_i - \mu) \right) = -\frac{1}{2} tr \left(\Sigma^{-1} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \right) \tag{54}$$

Et on réécrit la fonction de log-vraisemblance en séparant les deux sommes :

$$\log L(\mu, \Sigma) = -\frac{n}{2} \log(4\pi^2) + -\frac{n}{2} \log(\det(\Sigma)) - \frac{1}{2} tr \left(\Sigma^{-1} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \right) \tag{55}$$

Nous pouvons maintenant attaquer la dérivation partielle selon Σ.
Le premier terme, $-\frac{n}{2} \log(4\pi^2)$, est clairement constant ; sa dérivée vaut donc 0.
Pour le reste, on a :

$$\begin{aligned} \frac{\partial}{\partial \Sigma} \log L(\mu, \Sigma) &= -\frac{n}{2} tr(\Sigma^{-1}) - \frac{1}{2} tr \left(-\Sigma^{-1} \cdot \Sigma^{-1} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \right) \\ &= -\frac{1}{2} tr \left(\Sigma^{-1} \cdot \left(n I_2 - \Sigma^{-1} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \right) \right) \end{aligned} \tag{56}$$

Pour trouver le maximum de la fonction de log-vraisemblance, on doit trouver Σ tel que la dérivée partielle vaille 0. Puisque Σ^{-1} ne peut pas être nul, on a nécessairement :

$$n I_2 - \Sigma^{-1} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \ = 0 \tag{57}$$

En passant la deuxième partie à droite, en divisant le tout par n et en multipliant par Σ à gauche, on trouve notre estimateur de vraisemblance :

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \tag{58}$$

2 Etude Numérique

2.1 Préliminaires

2.1.1 Introduction

Dans le cadre de cette étude numérique, notre objectif est de représenter un échantillon de points tirés selon la loi de Z, ainsi que les ellipses d'isodensité associées à des probabilités p choisies.

Vous trouverez en pièce jointe le script python contenant toutes les fonction utilisées.

2.1.2 Importations

Nous commençons par importer les bibliothèques nécessaires :

```
import scipy.stats as stats
import matplotlib.pyplot as plt
import math
import numpy as np
from sympy import *
from mpl_toolkits.mplot3d import Axes3D
```

2.1.3 Utilitaires

Nous créons également trois fonctions utiles pour le reste du programme. La fonction generate points est notamment utile pour conserver le même ensemble de points, et faire en sorte que les graphiques soient tous cohérents entre eux.

```
def diagonaliser_matrice_numpy(matrice):
    # Calcul des valeurs propres et des vecteurs propres
    valeurs_propres, vecteurs_propres = np.linalg.eig(matrice)

    # Tri des valeurs propres et des vecteurs propres
    indices_tri = np.argsort(valeurs_propres)
    valeurs_propres_triees = valeurs_propres[indices_tri]
    vecteurs_propres_tries = vecteurs_propres[:, indices_tri]

    # Renvoie des valeurs propres, des vecteurs propres et de la matrice de passage

    return valeurs_propres_triees, vecteurs_propres_tries

def detMat(Matrice):
    return (Matrice[0][0]*Matrice[1][1]-Matrice[0][1]*Matrice[1][0])

def generatePoints(matriceCov,mu,p):
    x = stats.multivariate_normal.rvs(mu,Sigma ,1000)
    return x
```

2.1.4 Valeurs utilisées

Voici comment sont initalisées les variables :

```
mu = [0 ,0]
Sigma = [[0.25 ,0.3] ,[0.3 ,1.0]]
p = [0.1,0.5,0.8, 0.9]
x=generatePoints(Sigma,mu,p)
```

Pour que les figures illustratives soient cohérentes entre elles, nous utiliseront toujours le même ensemble de points, grâce à la fonction generatePoints.

2.1.5 Première représentation de la fonction à densité

Avant de commencer l'étude, regardons à quoi ressemble notre fonction à densité. Grâce à calculF et drawfX, on trace en 3D une représentation de la fonction à densité :

Par la suite, pour plus de visibilité, nous utiliserons un set de 1000 points, et nous tracerons les graphiques en 2D.

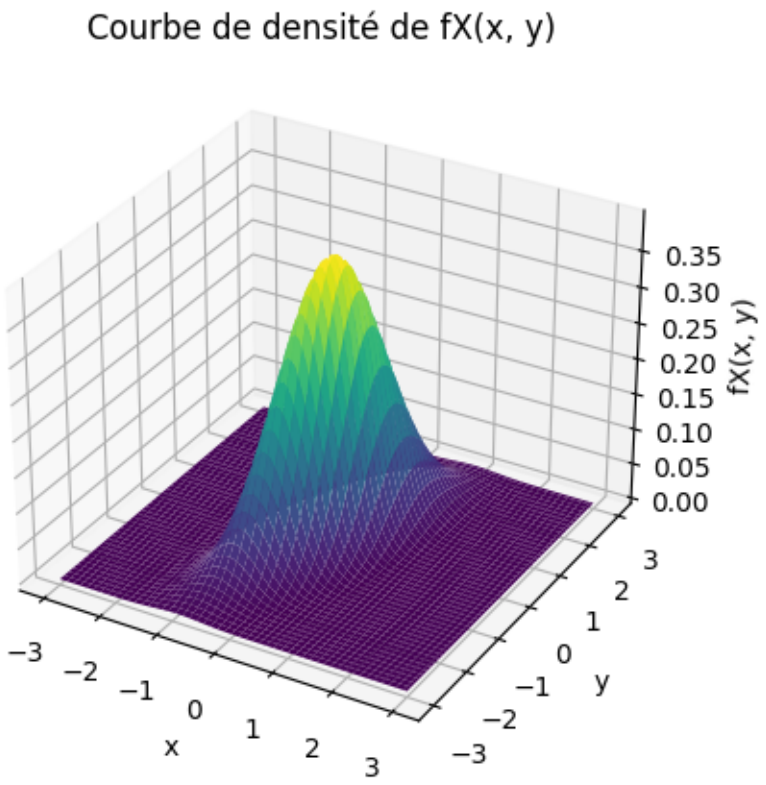

FIGURE 1 – Un échantillon de points tirés selon la loi de Z

2.2 Question 2.2.1

Ici, nous allons essayer de tracer des ellipses en fonction de différentes valeurs de p, et donc différetes valeurs de K.

La fonction principale est `drawPoints`, qui prend en paramètres la matrice de covariance (`matriceCov`), la moyenne (`mu`) et les probabilités (`p`) :

```
def drawPoints(matriceCov, mu, p,x):

    plt.scatter(x[:, 0], x[:, 1], s=5, edgecolors="black")
    # On s'attaque à l'ellipse
    ValPropres,VectPropres=diagonaliser_matrice_numpy(matriceCov)

    for element in p :

        k = (1-element)/(2*np.pi*math.sqrt(ValPropres[0] * ValPropres[1]))

        print(k)

        if(k>0):
            a= math.sqrt(ValPropres[0]*-2*math.log(k*2*np.pi*math.sqrt(ValPropres[0]*ValPropres[1])))
            b= math.sqrt(ValPropres[1]*-2*math.log(k*2*np.pi*math.sqrt(ValPropres[0]*ValPropres[1])))
            theta= math.atan(VectPropres[1][0]/VectPropres[1][1])
        else:
            print(k + 'n est pas valide')
```

```
        t = np.linspace(0, 2*np.pi, 100)
        ellipse_x = mu[0] + a * np.cos(t) * np.cos(theta) - b * np.sin(t) * np.sin(theta)
        ellipse_y = mu[1] + a * np.cos(t) * np.sin(theta) + b * np.sin(t) * np.cos(theta)

        plt.plot(ellipse_x, ellipse_y)

    plt.xlabel('Axe x')
    plt.ylabel('Axe y')
    plt.title('Ellipse')
    plt.grid(True)
    plt.axis('equal')
    plt.show()
```

Ensuite, nous appelons DrawPoints avec les paramètres définis plus tôt. On a le graphique ci-dessous :

2.2.1 Conclusion

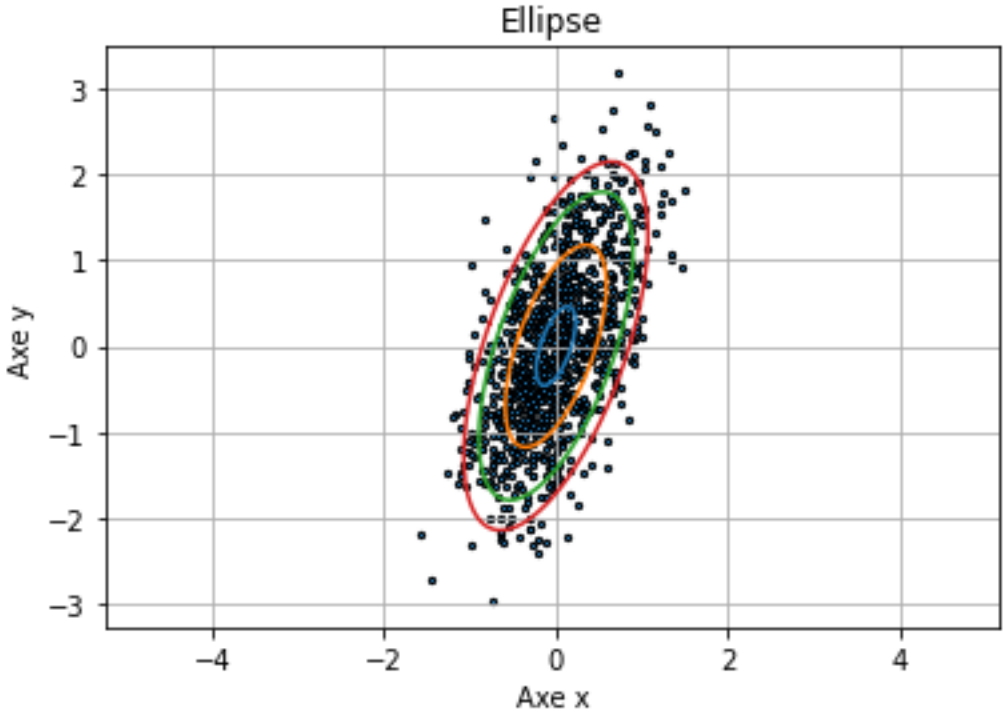

FIGURE 2 – Un échantillon de points tirés selon la loi de Z

2.2.2 Le nombre de points dans l'ellipse

Pour connaitre le nombre de points effectivement dans l'ellipse, on utilise la fonction suivante :

```
def calculatePointsInEllipses(Sigma, mu, p,x):
matriceCov = np.array(Sigma)  # Convertir Sigma en matrice numpy

points_in_ellipses = []

for element in p:
    k = (1 - element) / (2 * np.pi * math.sqrt(matriceCov[0, 0] * matriceCov[1, 1]))

    if k > 0:
        a = math.sqrt(matriceCov[0, 0] * -2 * math.log(k * 2 * np.pi * math.sqrt(matriceCov[0, 0]
        b = math.sqrt(matriceCov[1, 1] * -2 * math.log(k * 2 * np.pi * math.sqrt(matriceCov[0, 0]
        theta = math.atan(matriceCov[1, 0] / matriceCov[1, 1])

        points_inside_ellipse = 0

        for point in x:
            x_diff = point[0] - mu[0]
            y_diff = point[1] - mu[1]
            rotated_x = x_diff * np.cos(theta) + y_diff * np.sin(theta)
            rotated_y = -x_diff * np.sin(theta) + y_diff * np.cos(theta)
            ellipse_term = (rotated_x / a) ** 2 + (rotated_y / b) ** 2

            if ellipse_term <= 1:
                points_inside_ellipse += 1

        points_in_ellipses.append(points_inside_ellipse)

    else:
        print(str(k) + " n'est pas valide.")

for i in range(len(p)):
    print("Nombre de points dans l'ellipse", i+1, ":", points_in_ellipses[i])

return points_in_ellipses
```

Dans notre situation, on a les résultats suivants :

```
   Nombre de points dans l'ellipse 1 : 278
Nombre de points dans l'ellipse 2 : 727
Nombre de points dans l'ellipse 3 : 899
Nombre de points dans l'ellipse 4 : 946
[278, 727, 899, 946]
```

Les resultats sont assez proches des probabilités estimées.

2.2.3 Conclusion

Cette étude numérique a permis de générer un échantillon de points tirés selon la loi de Z et de représenter les ellipses d'isodensité associées à différentes probabilités p. Cette approche graphique permet de visualiser la distribution des points et les zones d'isodensité, ce qui facilite l'analyse des caractéristiques de la distribution.

2.3 Question 2.2(a)

2.3.1 Introduction

L'analyse des distributions de points est une tâche courante en statistiques et en data science. Elle permet de comprendre la répartition des données dans un espace multidimensionnel et de mettre en évidence des patterns ou des structures. Dans ce rapport, nous étudions la distribution de points tirés selon la loi de Z et nous visualisons les ellipses d'isodensité correspondantes pour différentes probabilités.

2.3.2 Méthode

Nous commençons par importer les bibliothèques nécessaires :

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

Ensuite, nous définissons la fonction `drawMeans`, qui prend en paramètres la matrice de covariance (`matriceCov`), la moyenne (`mu`) et les probabilités (`p`). Cette fonction génère un échantillon de points selon la loi de Z et trace les ellipses d'isodensité correspondantes :

```
def drawMeans(matriceCov, mu, p):
    x = stats.multivariate_normal.rvs(mu, matriceCov, 1000)
    plt.scatter(x[:, 0], x[:, 1], s=5, edgecolors="black")

    # Diviser les points en paquets
    num_points = [10, 50, 100]
    colors = ['red', 'green', 'blue']
    markers = ['x', '+', 'o']
    labels = ['Paquet de 10 points', 'Paquet de 50 points', 'Paquet de 100 points']

    for i, num in enumerate(num_points):
        num_paquets = 1000 // num
        paquets = np.split(x[:num_paquets * num], num_paquets)

        moyennes_x = np.mean(np.array(paquets)[:, :, 0], axis=1)
        moyennes_y = np.mean(np.array(paquets)[:, :, 1], axis=1)

        plt.scatter(moyennes_x, moyennes_y, marker=markers[i], color=colors[i], label=labels[i])

        # Calcul des valeurs propres et vecteurs propres de la matrice de covariance
        val_propres, vect_propres = np.linalg.eig(matriceCov)
        order = val_propres.argsort()[::-1]
        val_propres = val_propres[order]
        vect_propres = vect_propres[:, order]

        # Calcul des demi-axes de l'ellipse
        demi_axe_x = np.sqrt(val_propres[0]) * np.sqrt(-2 * np.log(1 - p[i]))
        demi_axe_y = np.sqrt(val_propres[1]) * np.sqrt(-2 * np.log(1 - p[i]))

        # Calcul de l'angle d'inclinaison de l'ellipse
        theta = np.degrees(np.arctan2(vect_propres[1, 0], vect_propres[0, 0]))

        # Tracer l'ellipse d'isodensité pour le groupe de moyennes
        t = np.linspace(0, 2 * np.pi, 100)
        ellipse_x = mu[0] + demi_axe_x * np.cos(np.radians(theta)) * np.cos(t) - demi_axe_y * np.sin(
        ellipse_y = mu[1] + demi_axe_x * np.sin(np.radians(theta)) * np.cos(t) + demi_axe_y * np.cos(

        plt.plot(ellipse_x, ellipse_y, color=colors[i])

    plt.scatter(mu[0], mu[1], color='pink', label='Coordonnées de mu')

    plt.xlabel('Axe x')
    plt.ylabel('Axe y')
    plt.title('Moyennes des paquets de points')
    plt.grid(True)
    plt.xlim(-5.5, 5.5)  # Définir la limite de l'axe x
    plt.ylim(-5.5, 5.5)
    plt.axis('equal')
    plt.legend(loc='best')
    plt.show()
```

Enfin, nous définissons les paramètres de la distribution de la loi normale bidimensionnelle, tels que la moyenne (`mu`) et la matrice de covariance (`Sigma`). Nous choisissons également les probabilités (`p`) pour lesquelles nous souhaitons tracer les ellipses d'isodensité :

```
mu = [0, 0]
Sigma = [[0.25, 0.3], [0.3, 1.0]]
p = [0.1, 0.5, 0.8, 0.9]
```

Ensuite, nous appelons la fonction `drawMeans` avec les paramètres appropriés pour générer le graphique contenant l'échantillon de points, les moyennes des paquets de points et les ellipses d'isodensité correspondantes :

```
drawMeans(Sigma, mu, p)
```

2.3.3 Résultats

Les résultats obtenus sont représentés dans la Figure 2. L'échantillon de points tirés selon la loi de Z est affiché en utilisant des points individuels. Les moyennes des paquets de points sont représentées par des symboles différents en fonction du nombre de points dans chaque paquet (10, 50 ou 100). Les ellipses d'isodensité sont tracées autour des groupes de moyennes pour différentes probabilités p.

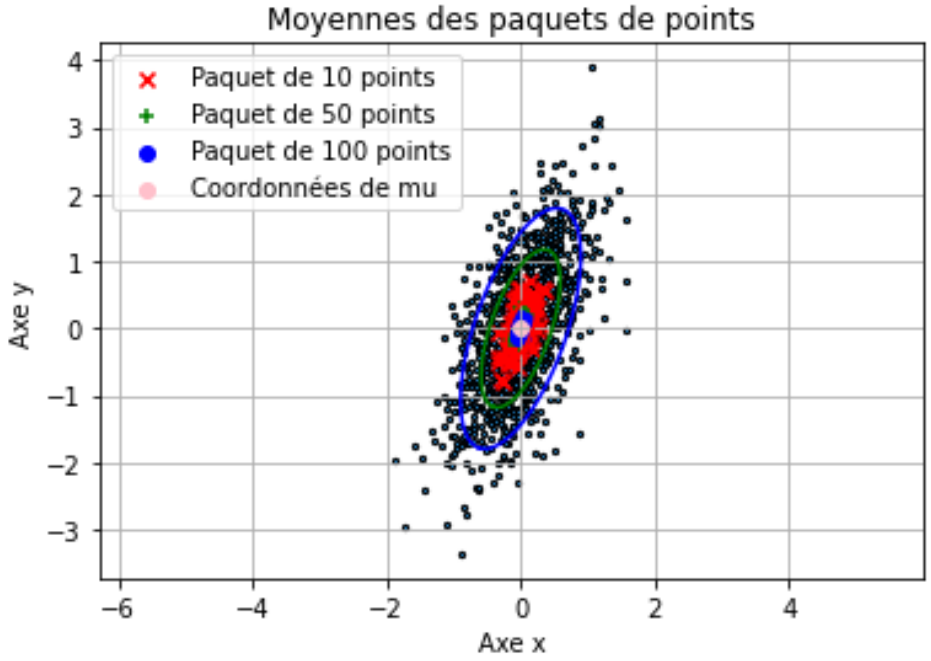

FIGURE 3 – Distribution des points et ellipses d'isodensité

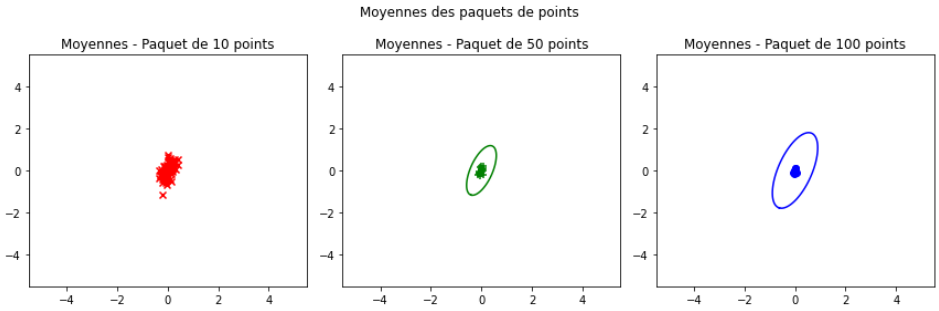

FIGURE 4 – Distribution des points et ellipses d'isodensité

2.3.4 Conclusion

Cette étude numérique a permis de générer un échantillon de points tirés selon la loi de Z, de calculer les moyennes des paquets de points et de représenter les ellipses d'isodensité associées à différentes probabilités p. Les résultats obtenus ont été tracés avec succès en utilisant les bibliothèques Python *scipy.stats* et *matplotlib.pyplot*. Cette approche graphique permet de visualiser la distribution des points, les regroupements de moyennes et les zones d'isodensité, facilitant ainsi l'analyse des caractéristiques de la distribution.

2.4 Question 2 (b)

2.4.1 Introduction

Le présent rapport porte sur l'estimation des paramètres d'une distribution multivariée, notamment la moyenne et la covariance, à partir d'échantillons de taille variable. L'objectif est d'illustrer la convergence de ces estimateurs en utilisant des ellipses d'isodensité. Ces ellipses représentent les régions de confiance autour des moyennes, indiquant ainsi la dispersion des données et les relations entre les variables.

Le code présenté dans ce rapport permet de calculer les estimateurs de la moyenne et de la covariance à partir d'échantillons de différentes tailles, et de visualiser leur convergence en traçant les ellipses d'isodensité correspondantes. Cette approche nous permet d'observer comment les estimations s'améliorent à mesure que la taille de l'échantillon augmente, reflétant ainsi une meilleure précision dans la représentation de la distribution sous-jacente.

2.4.2 Méthode

Cette partie du code, il génère un échantillon de points en utilisant la distribution multivariée normale avec une moyenne mu et une matrice de covariance matriceCov. Les points sont ensuite stockés dans la variable x.

```
def drawCovariance(matriceCov, mu, p):
x = stats.multivariate_normal.rvs(mu, matriceCov, 1000)
```

$Ici, nous définissons les différentes tailles de paquets de points que nous allons utiliser pour estimer la covariance. Les variable$
$graphiques, correspondant aux moyennes et aux covariances.$

```
num_points = [10, 50, 100]
colors = ['red', 'green', 'blue']
markers = ['x', '+', 'o']
labels = ['Paquet de 10 points', 'Paquet de 50 points', 'Paquet de 100 points']
fig, axs = plt.subplots(2, len(num_points), figsize=(12, 6))
```

Dans cette boucle, nous itérons sur les différentes tailles de paquets de points. Nous divisons l'échantillon de points en paquets de taille num et calculons les covariances de chaque paquet. Nous calculons également les moyennes des coordonnées x et y de chaque paquet. Les moyennes sont ensuite tracées sur le graphique supérieur (moyennes) correspondant à la taille de paquet actuelle.

```
for i, num in enumerate(num_points):
num_paquets = 1000 // num
paquets = np.split(x[:num_paquets * num], num_paquets)

covariances = np.zeros((num_paquets, 2, 2))
for j, paquet in enumerate(paquets):
    covariances[j] = np.cov(paquet.T)

moyennes_x = np.mean(np.array(paquets)[:, :, 0], axis=1)
moyennes_y = np.mean(np.array(paquets)[:, :, 1], axis=1)

axs[0, i].scatter(moyennes_x, moyennes_y, marker=markers[i], color=colors[i], label=labels[i]
axs[0, i].set_xlim(-5.5, 5.5)
```

```
axs[0, i].set_ylim(-5.5, 5.5)
axs[0, i].set_title('Moyennes - ' + labels[i])
```

Dans cette partie, nous calculons les valeurs propres et les vecteurs propres de la matrice de covariance initiale. Les valeurs propres sont triées par ordre décroissant et les vecteurs propres sont réarrangés en conséquence. En utilisant ces valeurs propres et vecteurs propres, nous calculons les demi-axes et l'orientation de l'ellipse correspondant à la probabilité de confiance p[i]. Nous générons ensuite les coordonnées de l'ellipse à partir de ces paramètres et les traçons sur le graphique supérieur.

```
val_propres, vect_propres = np.linalg.eig(matriceCov)
order = val_propres.argsort()[::-1]
val_propres = val_propres[order]
vect_propres = vect_propres[:, order]

demi_axe_x = np.sqrt(val_propres[0]) * np.sqrt(-2 * np.log(1 - p[i]))        demi_axe_y = np.sqrt

theta = np.degrees(np.arctan2(vect_propres[1, 0], vect_propres[0, 0]))

t = np.linspace(0, 2 * np.pi, 100)
ellipse_x = mu[0] + demi_axe_x * np.cos(np.radians(theta)) * np.cos(t) -
demi_axe_y * np.sin(np.radians(theta)) * np.sin(t)
ellipse_y = mu[1] + demi_axe_x * np.sin(np.radians(theta)) * np.cos(t) +
demi_axe_y * np.cos(np.radians(theta)) * np.sin(t)

axs[0, i].plot(ellipse_x, ellipse_y, color=colors[i])
```

Dans cette partie, nous itérons sur chaque paquet de points et calculons les covariances pour chaque paquet. En utilisant les valeurs propres et vecteurs propres de chaque covariance, nous calculons les demi-axes et l'orientation de l'ellipse correspondant à la probabilité de confiance p[i]. Nous générons les coordonnées de l'ellipse à partir de ces paramètres et les traçons sur le graphique inférieur (covariances) correspondant à la taille de paquet actuelle. De plus, nous traçons également les moyennes des paquets de points sur le même graphique.

```
        for j in range(num_paquets):
        val_propres, vect_propres = np.linalg.eig(covariances[j])
        order = val_propres.argsort()[::-1]
        val_propres = val_propres[order]
        vect_propres = vect_propres[:, order]

        demi_axe_x = np.sqrt(val_propres[0]) * np.sqrt(-2 * np.log(1 - p[i]))
        demi_axe_y = np.sqrt(val_propres[1]) * np.sqrt(-2 * np.log(1 - p[i]))

        theta = np.degrees(np.arctan2(vect_propres[1, 0], vect_propres[0, 0]))

        ellipse_x = moyennes_x[j] + demi_axe_x * np.cos(np.radians(theta)) * np.cos(t) - demi_axe
        ellipse_y = moyennes_y[j] + demi_axe_x * np.sin(np.radians(theta)) * np.cos(t) + demi_axe

    axs[1, i].plot(ellipse_x, ellipse_y, color='black', alpha=0.2)
    axs[1, i].scatter(moyennes_x, moyennes_y, marker='.', color=colors[i], alpha=0.5)
    axs[1, i].set_xlim(-5.5, 5.5)
    axs[1, i].set_ylim(-5.5, 5.5)
    axs[1, i].set_title('Covariances - ' + labels[i])
```

Enfin, nous ajoutons un titre à la figure globale, améliorons la mise en page et affichons la figure. Nous définissons également les valeurs de mu (moyenne) et Sigma (matrice de covariance) ainsi que les probabilités de confiance p. En appelant la fonction drawCovariance avec ces paramètres, nous exécutons le code et visualisons les résultats.

```
    fig.suptitle('Moyennes et covariances des paquets de points')
    plt.tight_layout()
    plt.show()

mu = [0, 0]
Sigma = [[0.25, 0.3], [0.3, 1.0]]
p = [0.1, 0.5, 0.8, 0.9]
drawCovariance(Sigma, mu, p)
```

2.4.3 Résultat

Les graphiques obtenus illustrent la convergence des estimateurs de la moyenne et de la covariance avec l'augmentation de la taille des échantillons. Les ellipses d'isodensité représentent les régions de confiance autour des moyennes, montrant ainsi comment la dispersion des données diminue et la précision des estimations s'améliore à mesure que la taille des échantillons augmente.

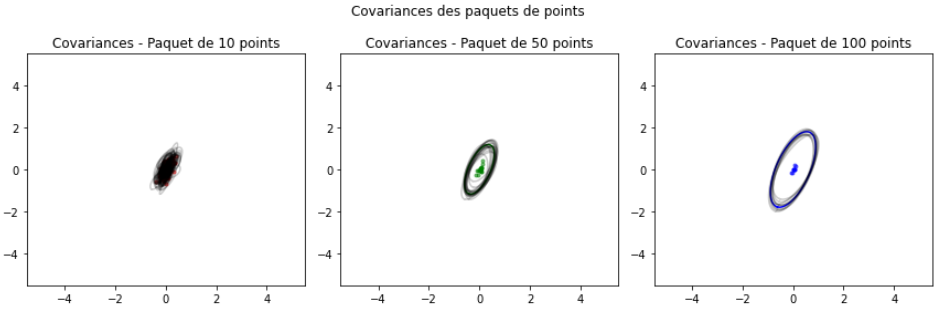

FIGURE 5 – Convergence des estimateurs de la moyenne et de la covariance

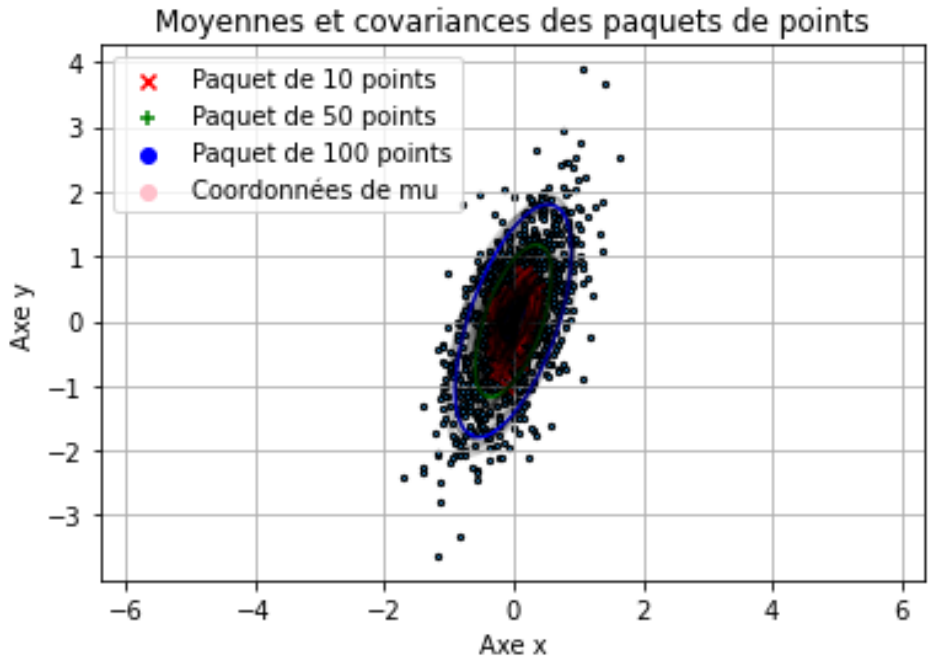

FIGURE 6 – Convergence des estimateurs de la moyenne et de la covariance (Version superposé)

2.4.4 Conclusion

En conclusion, nous avons présenté un code permettant de calculer et de visualiser les estimateurs de la moyenne et de la covariance à partir d'échantillons de taille variable. Les ellipses d'isodensité tracées nous ont permis de comprendre comment les estimations convergent vers les vraies valeurs à mesure que la taille de l'échantillon augmente. Cette méthode offre une représentation visuelle claire de la convergence des estimateurs et démontre l'importance de disposer de données plus nombreuses pour des estimations plus précises.

En utilisant ce code et en comprenant les concepts sous-jacents, il est possible d'appliquer cette approche à d'autres distributions multivariées et d'explorer davantage les propriétés des estimateurs.

3 Conclusion general

L'étude numérique réalisée a permis de générer un échantillon de points tirés selon la loi de Z et d'analyser les caractéristiques de cette distribution en utilisant les ellipses d'isodensité associées à des probabilités p choisies. Grâce à l'utilisation des bibliothèques Python *scipy.stats* et *matplotlib.pyplot*, nous avons pu tracer avec succès ces ellipses et visualiser la distribution des points ainsi que les zones d'isodensité.

Dans un premier temps, nous avons représenté graphiquement l'échantillon de points tirés selon la loi de Z et les ellipses d'isodensité correspondantes. Cette approche graphique nous a permis de mieux comprendre la répartition des points et d'identifier les zones de densité plus élevée. Cette représentation visuelle nous offre une première indication sur les caractéristiques de la distribution.

Ensuite, nous avons calculé les estimateurs de la moyenne et de la covariance à partir d'échantillons de taille variable. Nous avons observé la convergence de ces estimateurs en utilisant les ellipses d'isodensité pour différentes valeurs de probabilité p. Les résultats ont montré que les estimations s'améliorent à mesure que la taille de l'échantillon augmente, ce qui reflète une meilleure précision dans la représentation de la distribution sous-jacente.

L'analyse des regroupements de moyennes et des ellipses d'isodensité a permis de mettre en évidence la manière dont les estimations évoluent avec la taille de l'échantillon. Les ellipses d'isodensité ont révélé une diminution de la dispersion des données et une meilleure précision des estimations lorsque la taille de l'échantillon augmentait. Cela souligne l'importance d'avoir des données plus nombreuses pour obtenir des estimations plus fiables et précises.

En conclusion, cette étude numérique nous a permis de comprendre l'importance de l'échantillonnage et de la taille de l'échantillon dans l'estimation des paramètres d'une distribution multivariée. Les ellipses d'isodensité se sont révélées être un outil graphique efficace pour visualiser la convergence des estimateurs de la moyenne et de la covariance. Cette approche offre une représentation visuelle claire et concise de la manière dont les estimations s'améliorent à mesure que la taille de l'échantillon augmente. En appliquant ces concepts et cette méthode à d'autres distributions multivariées, il est possible d'explorer davantage les propriétés des estimateurs et d'améliorer nos connaissances en statistique multivariée.