Exercice

Cyrille KAMGA

29 mai 2022

Introduction

On considère l'espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ sur lequel les variables aléatoires suivantes sont définies :

$$Z_i = \sqrt{\rho}X + \sqrt{1-\rho}\varepsilon_i \tag{1}$$

avec, $i \in \{1, ..., N\}$ et $X \sim \mathcal{N}(0,1)$ et $(\varepsilon_i)_i \sim \mathcal{N}(0,1)$ qui sont toutes indépendantes.

Ces variables aléatoires représentent les variables latentes servant à la modélisation du défaut des contreparties. On dira qu'une entreprise i entre en défaut lorsque sa variable latente descend en dessous d'un certain seuil.

On note, $n_i^0 \in \{0,1,2,3\}$ la note initiale de l'entrprise i et n_i sa note aléatoire dans 1 an. Le profil de risque d'une entreprise est régi par une matrice de transition à 1 an notée M et sa cumulée sur les lignes par C.

Par ailleurs on a la relation suivante :

$$\{n_i = k\} = \{C_{n_i^0, k-1} \leq 1 - \Phi(Z_i) \leq C_{n_i^0, k}\} \tag{2}$$

avec $C_{m,-1} = 0$ pour tout m et Φ la fonction de répartition d'une loi normale centrée et réduite.

Question 1.
Pour $k \in \{0,1,2,3\}$ et $n_i^0 \in \{0,1,2,3\}$, donnez $\mathbb{P}(n_i = k | n_i^0)$

Soit $A_j^i = \{n_i^0 = j\}$, les $(A_j^i)_{j \in \{0,1,2,3\}}$ forment une partition de Ω et $\mathcal{B} = \sigma(A_j^i, j \in \{0,1,2,3\})$. Ainsi, d'après la formule de l'espérance conditionnelle dans le cas discret on a :

$$\begin{aligned}
\mathbb{P}(n_i = k | n_i^0) &= \mathbb{E}(\mathbb{1}_{\{n_i = k\}} | n_i^0) \\
&= \sum_{j \in J, t.q., \mathbb{P}(A_j^i) > 0} \mathbb{E}(\mathbb{1}_{\{n_i = k\}} | n_i^0 = j) \mathbb{1}_{\{n_i^0 = j\}} \\
&= \sum_{j \in J, t.q., \mathbb{P}(A_j^i) > 0} \mathbb{P}(n_i = k | n_i^0 = j) \mathbb{1}_{\{n_i^0 = j\}} \\
&= \sum_{j \in J, t.q., \mathbb{P}(A_j^i) > 0} M_{j,k} \mathbb{1}_{\{n_i^0 = j\}}
\end{aligned}$$

avec $J = 0,1,2,3$ et $M_{j,k}$ la probabilité pour une entreprise d'être notée k en étant initialement notée j.

Question 2.
Pour $k \in \{0,1,2,3\}$ et $n_i^0 \in \{0,1,2,3\}$, calculez $\mathbb{P}(n_i = k | n_i^0, X)$

Soit une fonction h borélienne telle que $h(n_i^0, X)$ soit $\sigma(n_i^0, X)$-mesurable, une telle fonction existe d'après la définition de la mesurabilité.

En posant $Z = (n_i^0, X)$, et utilisant la relation 2, on a :

$$\begin{aligned}
\mathbb{P}(n_i = k|Z) &= \mathbb{E}(\mathbb{1}_{\{n_i=k\}}|Z) \\
&= \mathbb{E}(\mathbb{1}_{\{C_{n_i^0,k-1} \leq 1-\Phi(Z_i) \leq C_{n_i^0,k}\}}|Z) \\
&= \mathbb{E}(\mathbb{1}_{\{C_{n_i^0,k-1} \leq 1-\Phi(\sqrt{\rho}X+\sqrt{1-\rho}\varepsilon_i) \leq C_{n_i^0,k}\}}|Z) \\
&= \mathbb{E}(f(\varepsilon_i, Z)|Z)
\end{aligned}$$

où $f : \mathbb{R} \times \{0,1,2,3\} \times \mathbb{R} \to \{0,1\}$, définit par $f(y,j,x) = \mathbb{1}_{\{C_{j,k-1} \leq 1-\Phi(\sqrt{\rho}x+\sqrt{1-\rho}y) \leq C_{j,k}\}}$. f est borélienne et $f \in L^1$.

Comme ε_i et X sont indépendantes, et en considérant que le risque *idiosyncratique* ne dépend pas de la note initiale n_i^0 (ce qui est une hypothèse plutôt réaliste en pratique), on a d'après les proprietés de l'espérance conditionnelle :

$$\mathbb{E}(f(\varepsilon_i, Z)|Z) = g(Z),$$

où g est défini pour tout $z = (j, x)$, dans $\{0,1,2,3\} \times \mathbb{R}$, par $g(z) = \mathbb{E}(f(\varepsilon_i, z))$

En effet, on a :

— (*i*) g(Z) qui est bien $\sigma(n_i^0, X)$-mesurable car dépend de Z

— (*ii*) Soit h mesurable et borné, alors $h(Z)$ est $\sigma(n_i^0, X)$-mesurable. En utilisant le théorème de Fubini on a :

$$\begin{aligned}
\mathbb{E}(f(\varepsilon_i, Z)h(Z)) &= \int f(y,z)h(z)dP_{\varepsilon_i} \otimes dP_Z(y,z) \\
&= \int h(z) \left(\int f(y,z)dP_{\varepsilon_i} \right) dP_Z(z) \\
&= \int h(z)\mathbb{E}(f(\varepsilon_i, z))dP_Z(z) \\
&= \mathbb{E}(h(Z)g(Z))
\end{aligned}$$

D'où d'après la définition de l'espérance conditionnelle on a $g(Z) = \mathbb{E}(f(\varepsilon_i, Z)|Z)$

En utilisant le fait que $\Phi(-a) = 1 - \Phi(a)$, et comme Φ est continue et strictement croissante on note par Φ^{-1} son inverse (au sens de la bijection), nous avons alors,

$$\begin{aligned}
\mathbb{E}(f(\varepsilon_i, z)) &= \mathbb{P}(C_{j,k-1} \leq 1 - \Phi(\sqrt{\rho}x + \sqrt{1-\rho}\varepsilon_i) \leq C_{j,k})) \\
&= \mathbb{P}(C_{j,k-1} \leq \Phi(-\sqrt{\rho}x - \sqrt{1-\rho}\varepsilon_i) \leq C_{j,k})) \\
&= \mathbb{P}(\Phi^{-1}(C_{j,k-1}) \leq -\sqrt{\rho}x - \sqrt{1-\rho}\varepsilon_i \leq \Phi^{-1}(C_{j,k})) \\
&= \mathbb{P}(\frac{\Phi^{-1}(C_{j,k-1}) + \sqrt{\rho}x}{\sqrt{1-\rho}} \leq -\varepsilon_i \leq \frac{\Phi^{-1}(C_{j,k}) + \sqrt{\rho}x}{\sqrt{1-\rho}})
\end{aligned}$$

Or comme $\varepsilon_i \sim \mathcal{N}(0,1)$, on a par symétrie de la loi normale $-\varepsilon_i \sim \mathcal{N}(0,1)$. Nous obtenons ainsi,

$$\begin{aligned}
\mathbb{E}(f(\varepsilon_i, z)) &= \mathbb{P}\left(\frac{\Phi^{-1}(C_{j,k-1}) + \sqrt{\rho}x}{\sqrt{1-\rho}} \leq \varepsilon_i \leq \frac{\Phi^{-1}(C_{j,k}) + \sqrt{\rho}x}{\sqrt{1-\rho}} \right) \\
&= \Phi\left(\frac{\Phi^{-1}(C_{j,k}) + \sqrt{\rho}x}{\sqrt{1-\rho}} \right) - \Phi\left(\frac{\Phi^{-1}(C_{j,k-1}) + \sqrt{\rho}x}{\sqrt{1-\rho}} \right)
\end{aligned}$$

En somme toute, on a donc :

$$\mathbb{P}(n_i = k|n_i^0, X) = \mathbb{E}(f(\varepsilon_i, Z)) = \Phi\left(\frac{\Phi^{-1}(C_{n_i^0,k}) + \sqrt{\rho}X}{\sqrt{1-\rho}}\right) - \Phi\left(\frac{\Phi^{-1}(C_{n_i^0,k-1}) + \sqrt{\rho}X}{\sqrt{1-\rho}}\right)$$

Question 3.
Développez des fonctions en Python permettant de simuler l'ensemble des notes de toutes les entreprises conditionnellement à un facteur X simulé en amont.

pour simuler l'ensemble des notes de toutes les entreprises conditionnellement à X, nous allons utiliser la méthode de la fonction inverse. La donnée d'un générateur aléatoire uniforme permet, en théorie, de simuler n'importe quelle distribution à partir de l'inverse de sa fonction de répartition.

Posons $Y = \mathbb{P}(n_i = k|n_i^0, X)$, Y est bien $\sigma(n_i^0, X)$-mesurable, notons par F_Y sa fonction de répartition. Son inverse généralisé (ou fonction quantile) est définie pour tout $u \in [0,1]$ par :

$$F_Y^{\leftarrow}(u) = inf\{k \in \mathbb{R} : F_Y(k) \geq u\} \tag{3}$$

Y est une variable aléatoire discrète de support $\Omega = \{k, k \in \{0,1,2,3\}\}$ fini. Notons pour $k \in \{0,1,2,3\}$,

$$p_k = \mathbb{P}(Y = k) \text{ et } c_k = \textstyle\sum_{i=0}^{k} p_k$$

On peut donc réécrire l'inverse généralisé de Y, donné en 3,

$$F_Y^{\leftarrow}(u) = \{k \in \{0,1,2,3\} : c_{k-1} \leq u \leq c_k\} \text{ avec } c_{-1} = 0$$

Ainsi la variable aléatoire N définie comme suite, avec $u \sim \mathcal{U}([0,1])$ suit la même loi que Y

$$N = \begin{cases} 0 & \text{si } u \in [0, s_0] \\ k & \text{si } u \in]s_{k-1}, s_k], k \geq 1 \end{cases}$$

Programmation :

Le programme se découpera en deux principales fonctions :

1. Une première fonction qui calcule la $\mathbb{P}(n_i = k|n_i^0 = j, X = x)$ pour des valeurs de j et x données
2. Ensuite, une fonction qui génère, par la méthode de la fonction inverse, des réalisations k, suivant la loi de n_i (note aléatoire de l'entreprise i) conditionnellement à un facteur X simulé en amont et de la note initiale n_i^0.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.random import random_sample
from scipy.stats import norm
```

Pour la première fonction nous allons utiliser les fonctions, *cdf* et *ppf* du package *scipy.stats* pour calculer respectivement la fonction de répartition et la fonction de répartition inverse d'une loi normale centrée et réduite.

[2]:
```
def probaCond_notation(k, n0, X, rho) :
    M = np.matrix('0.7, 0.2, 0.05, 0.05; 0.15, 0.65, 0.1, 0.1; 0.05, 0.05, 0.7, 0.2;␣
↪0, 0, 0, 1')
    C = np.cumsum(M, 1)
    if k == 0 :
```

```
        c = 0
    elif (set([k, n0]).issubset(set([0, 1, 2, 3]))) :
        c = C[n0, k-1]
    else :
        return 0
    a = (norm.ppf(C[n0, k]) + np.sqrt(rho)*X)/np.sqrt(1-rho)
    b = (norm.ppf(c) + np.sqrt(rho)*X)/np.sqrt(1-rho)
    return norm.cdf(a) - norm.cdf(b)
```

Pour le générateur de l'ensemble des notes de toutes les entreprises conditionenellement à X, nous allons utiliser la fonction *digitize* de la librairie *numpy*. Cette fonction permet d'obtenir les indices des "bins" auxquels chaque valeur d'un tableau de loi de uniforme $[0, 1]$ appartient.

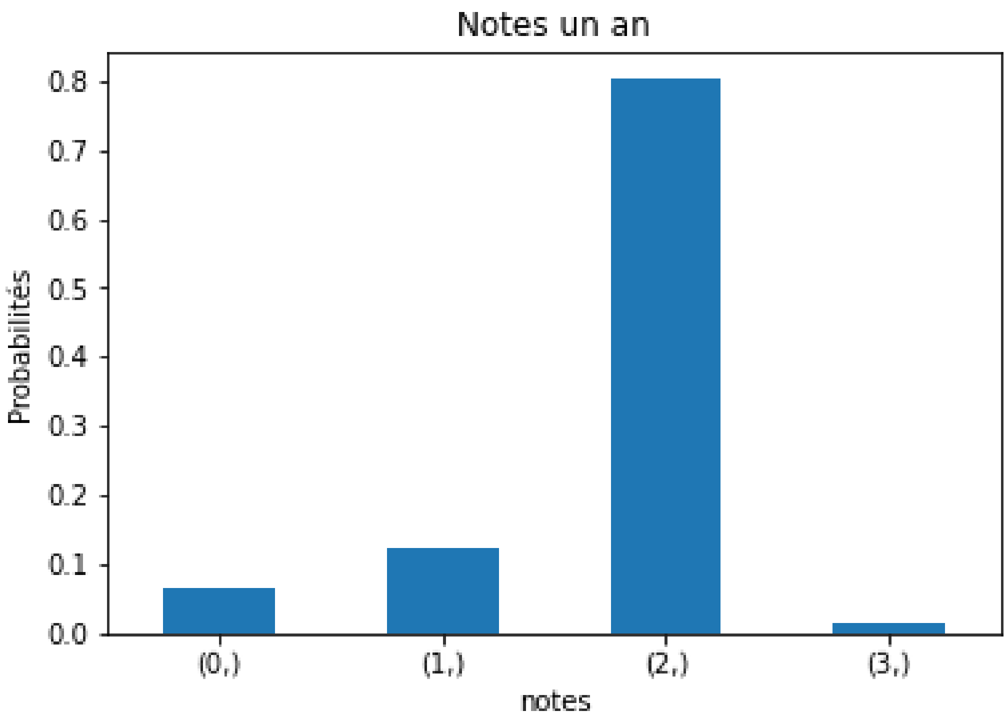
[4]:
```
def rv_note_cond (loi_n0, X, rho, N):
    sample = np.zeros(N)
    support = np.array([0, 1, 2, 3]) # Le support de la note aleatoire n_i

    if loi_n0 in [0, 1, 2, 3] :
        probabilites = [probaCond_notation(k, loi_n0, X, rho) for k in range(4)]
        sample = support[np.digitize(random_sample(N), np.cumsum(probabilites))]


    elif (callable(loi_n0)) and (loi_n0(1) in [0, 1, 2, 3]) :
        Mat_proba = np.matrix(np.reshape([probaCond_notation(k, j, X, rho) for k in␣
↪range(4) for j in range(3) ], (4,3)))
        n0 = loi_n0(N)
        u = random_sample(N)

        sample[np.where(n0 == 0)[0]] = support[np.digitize(u[np.where(n0 == 0)[0]], np.
↪cumsum([Mat_proba[:,0]]))]
        sample[np.where(n0  == 1)[0]] = support[np.digitize(u[np.where(n0  == 1)[0]],␣
↪np.cumsum([Mat_proba[:,1]]))]
        sample[np.where(n0  == 2)[0]] = support[np.digitize(u[np.where(n0  == 2)[0]],␣
↪np.cumsum([Mat_proba[:,2]]))]

    else :
        print('Loi note initiale inappropriée ou arguments incompatible')

    return sample
```

Exemple : Nous proposons de faire une petite application numérique en représentant l'histogramme de la distribution des notes de $N = 500$ entreprises qui partent toutes d'une note initiale $n_i^0 = 2$ pourtout i et en posant $\rho = 0.5$.

[26]:
```
def hist_note_UnAn(n0, rho, N):
    x = np.random.normal()
    Y = rv_note_cond (n0, x, rho, N)
    Y = pd.DataFrame(Y)
    a = Y.value_counts(sort=False)
    a = a/500
    a.plot.bar(rot=1)
    plt.title('Notes un an')
    plt.xlabel('notes')
```

```python
    plt.ylabel('Probabilités')
    plt.show()
```

[27]:
```python
n0, rho = 2,.5
N = 500
hist_note_UnAn(n0, rho, N)
```



Commentaires :

— Une entreprise qui part d'une note initiale $n_i^0 = 2$ à 80% de chances d'avoir la même note à un horizon d'un an, un peu plus de 10% d'avoir une note égale 1 dans un an, etc.

Question 4.

On considère maintenant la variable aléatoire $L = \sum_{i=1}^{N} E_i \mathbb{1}_{\{n_i=3\}}$, représentant la perte à un an, avec E_i de loi Log-Normale de paramètres $= 0$ et $\sigma = 1$ désignant l'exposition de l'entreprise i. Proposez une fonction réalisant un calcul par la méthode de Monte-Carlo du quantile d'ordre α de L

On pose,

$$\begin{cases} n_i^0 \sim \mathcal{U}(\{0,1,2\}) \\ N = 1000 \\ \rho = 0.5 \\ \alpha = 0.65 \end{cases}$$

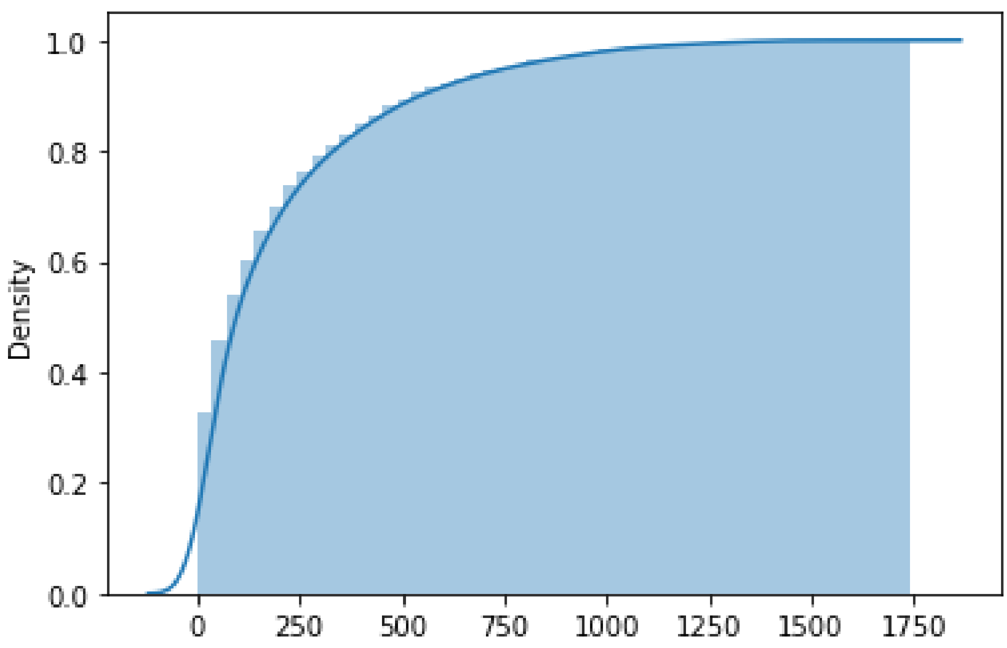
[20]:
```python
# Question 4
def simu_perte(loi_n0, rho, N, Nmc):
    X = np.random.normal(size = Nmc)
```

```
    E = np.random.lognormal(size=(Nmc,N))
    Mat_notes = np.matrix(np.reshape([rv_note_cond (loi_n0, X[k], rho, N) for k in␣
↪range(Nmc)], (Nmc,N)))
    Mat_1_0 = np.multiply(Mat_notes==np.full((Nmc, N), 3), 1)
    Product = np.mat(np.array(E) * np.array(Mat_1_0)) # Produit termes à termes

    return np.concatenate(np.sum(Product,axis=1).tolist())
```

Nous pouvons faire une représentation de la fonction de répartition de L :

[31]:
```
import seaborn as sns
x = simu_perte(loi_n0, rho, N, Nmc)
kwargs = {'cumulative': True}
sns.distplot(x, hist_kws=kwargs, kde_kws=kwargs)
```

[31]: <AxesSubplot:ylabel='Density'>



[21]:
```
def quantile_L(alpha, loi_n0, rho, N, Nmc):
    data = simu_perte(loi_n0, rho, N, Nmc)
    return np.quantile(data, alpha)
```

[22]:
```
def Uniforme_discrete(N):
    return np.random.randint(0, high = 3, size = N)
```

[28]:
```
alpha, rho = 0.65, 0.5
loi_n0 = Uniforme_discrete
N, Nmc = 1000, 10000
```

Ainsi, le quantile d'ordre $\alpha = 0.65$ est donné par,

[29]:
```
quantile_L(alpha, loi_n0, rho, N, Nmc)
```

[29]: 175.2178524758051

Question 5.

A partir des simuLations du défaut réalisées, proposez une méthodologie d'estimation paramétrique afin de retrouver une estimation du paramètre ρ. Posez simplement le problème à résoudre

---

A partir de la relation 1 nous remarquons que :

$$\begin{aligned} Cov(Z_i, Z_j) &= Cov(\sqrt{\rho}X + \sqrt{1-\rho}\varepsilon_i, \sqrt{\rho}X + \sqrt{1-\rho}\varepsilon_j) \\ &= Cov(\sqrt{\rho}X, \sqrt{\rho}X) + Cov(\sqrt{1-\rho}\varepsilon_i, \sqrt{1-\rho}\varepsilon_j) \\ &= \rho Var(X) = \rho \end{aligned}$$

En effet, les $(\varepsilon_i)_i$ et X sont *iid* donc $Cov(\sqrt{1-\rho}\varepsilon_i, \sqrt{1-\rho}\varepsilon_j) = 0$. De plus $X \sim \mathcal{N}(0, 1)$.

Comme $Cov(Z_i, Z_j) = \mathbb{E}(Z_i Z_j)$, une estimation de ρ par la méthode de Monte Carlo serait :

$$\hat{\rho} = \frac{1}{N} \sum_{i,j=1}^{N} X_i X_j$$

où l'échantillon (X_i, X_j) est simulé suivant la loi du couple (Z_i, Z_j).

Le problème est donc de trouver la loi jointe (Z_i, Z_j) connaissant juste les lois marginales Z_i et Z_j qui sont toutes les deux de loi $\mathcal{N}(0, 1)$.

Deux possibilités :

— On peut soit utiliser les données historiques pour estimer empiriquement ρ.

— Soit on utilise la méthode des copules gaussiènne pour la simulation de la loi jointe (Z_i, Z_j) et l'estimation de ρ.