École Nationale
Supérieure
de l'Électronique
et de ses Applications

Statistiques Multidimentionnelles et Inférentielles

Responsable du cours : Bastien Faucard
bastien.faucard@ensea.fr

E.N.S.E.A. 2024 – 2025, Semestre 8

Travaux Pratiques

Table des matières

1 TP1 : Pierre-feuille-ciseaux

Problématique

Pierre-feuille-ciseaux est un jeu opposant deux joueurs. À chaque tour, les joueurs annoncent simultanément le coup qu'ils ont choisi parmi les trois possibilités "Pierre", "Feuille" ou "Ciseaux". Les coups gagnants sont représentés dans la figure 1.1. Si les deux joueurs choisissent le même coup, la manche est nulle.

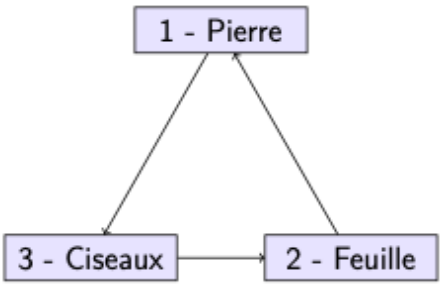

FIGURE 1.1 – Coups gagnants à Pierre-feuille-ciseaux

Une stratégie efficace pour ne pas perdre à ce jeu consiste à choisir ses coups de manière indépendante, et uniformément dans l'ensemble $\{\text{Pierre}, \text{Feuille}, \text{Ciseaux}\}$. Cependant, les humains éprouvent d'énormes difficultés à produire un véritable hasard : ils sont influencés par diverses considérations psychologiques [1]. Un ordinateur peut ainsi battre un humain, s'il arrive à anticiper les action s de son adversaire. Pour cela, il connaît l'historique (l'échantillon) des coups réalisés par son adversaire. À partir de cet échantillon, l'ordinateur peut inférer la loi de génération des coups de son adversaire humain. Il s'agit donc d'un problème d'estimation. L'objectif de ce TP est de construire une intelligence artificielle basée sur ces principes.

Modèle du comportement du joueur humain par chaîne de Markov

Nous allons dans un premier temps développer un modèle extrêmement rudimentaire. Nous allons respectivement utiliser les entiers $1, 2$ et 3 pour représenter les coups "Pierre", "Feuille" et "Ciseaux". On suppose que les choix de l'humain sont des réalisations d'un processus aléatoire $(X_k)_{k\geq 1}$ où

1. Par exemple, les hommes ont plus tendance à choisir le coup "Pierre" (en anglais *rock*, associé à une idée de virilité : "Rock is for rookies".

1. les X_k sont des variables aléatoires à valeurs dans $\{1,2,3\}$,
2. X_1 est distribuée selon la loi uniforme sur $\{1,2,3\}$,
3. pour tout $a_1,\dots,a_{n+1}\in\{1,2,3\}$, on a

$$P(X_{n+1}=a_{n+1}|X_1=a_1,\dots,X_n=a_n)=P(X_{n+1}=a_{n+1}|X_n=a_n),$$

4. pour tout $i,j\in\{1,2,3\}$, on a

$$P(X_{n+1}=j|X_n=j)=p_{i,j}$$

où les $p_{i,j}$ sont les coefficients d'une matrice

$$\mathscr{P}=\begin{pmatrix}p_{1,1}&p_{1,2}&p_{1,3}\\p_{2,1}&p_{2,2}&p_{2,3}\\p_{3,1}&p_{3,2}&p_{3,3}\end{pmatrix}$$

Un processus qui vérifie les conditions (3) et (4) est appelé *chaîne de Markov*. La condition (3) signifie que lorsque l'on cherche à prédire le coup futur X_{n+1}, l'information donnée par la connaissance du passé (les réalisation de $X_1,\dots,X_n$) n'est pas entièrement utilisée : seul l'état présent (la réalisation de X_n) permet de prédire l'état futur. La matrice $\mathscr{P}$ est appelée la *matrice de transition* du processus, $p_{i,j}$ représente la probabilité de choisir le coup j sachant que l'on avait précédemment choisi le coup i. Remarquons que les coefficients $p_{i,j}$ sont les paramètres inconnus de notre modèle.

1.1 Préparation

1.1.1 Estimation des paramètres du modèle

1. Donner une relation entre les colonnes de la matrice $\mathscr{P}$. En déduire que la loi du processus $(X_k)_{k\geq 1}$ est entièrement déterminée seulement par les six paramètres inconnus $p_{i,j}$, pour $i\in\{1,2,3\}$ et $j/in\{1,2\}$.
2. Montrer que pour tout $a_1,\dots,a_n\in\{1,2,3\}$,

$$P(X_1=a_1,\dots,X_n=a_n)=\frac{1}{3}\prod_{k=1}^{n-1}p_{a_k,a_{k+1}},$$

Indication : on pourra effectuer une récurrence sur $n\geq 1$.

3. On suppose que l'historique des coups passés est $(a_1,\dots,a_n)$. Pour $i,j\in\{1,2,3\}$, on note $n_{i,j}$ le nombre de transitions du coup i vers le coup j :

$$n_{i,j}=\mathrm{Card}\{k\in\{1,\dots,n-1\},(a_k,a_{k+1})=(i,j)\}$$

Montrer que la vraissemblance de $(X_1,\dots,X_n)$ est

$$\mathcal{L}_{X_1,\dots,X_n}(a_1,\dots,a_n,\mathscr{P})=\frac{1}{3}\prod_{i=1}^{3}\Big(p_{i,1}^{n_{i,1}}p_{i,2}^{n_{i,2}}(1-p_{i,1}-p_{i,2})^{n_{i,3}}\Big).$$

4. On appelle $(p'_{1,1},p'_{1,2},p'_{2,1},p'_{2,2},p'_{3,1},p'_{3,2})$ l'estimation du maximum de vraissemblance de $(p_{1,1},p_{1,2},p_{2,1},p_{2,2},p_{3,1},p_{3,2})$. En considérant la Log-vraissembance, montrer que pour tout $i\in\{1,2,3\}$ et $j\in\{1,2\}$

$$\frac{n_{i,j}}{p'_{i,j}}=\frac{n_{i,3}}{1-p'_{i,1}-p'_{i,2}}.$$

5. En déduire que pour tout $i,j\in\{1,2,3\}$, on a

$$p'_{i,j}=\frac{n_{i,j}}{n_{i,1}+n_{i,2}+n_{i,3}}.$$

1.1.2 Prise de décision

Grâce à la question 5, nous avons estimé les probabilités de transition à partir des n derniers coups du joueur humain. Notons i le dernier coup joué par l'humain (au tour n). L'ordinateur doit maintenant jouer au tour $n+1$ une valeur $k \in \{1,2,3\}$ que nous cherchons à déterminer. Il pourra gagner (relativement à son adversaire) 1, 0 ou -1 points.
Pour $k \in \{1,2,3\}$, on note $g(k)$ le gain relatif moyen réalisé par l'ordinateur.

6. Montrer que

$$\begin{cases} g(1) &= p_{i,3} - p_{i,2} \\ g(2) &= p_{i,1} - p_{i,3} \\ g(3) &= p_{i,2} - p_{i,1} \end{cases}$$

7. De manière idéale, l'ordinateur souhaiterait jouer au coup $n+1$ la valeur k en laquelle g admet un maximum. EN prenant en compte le fait que les paramètres $p_{i,j}$ sont inconnus, proposer une méthode de prise de décision.

1.2 Travail en séance

1.2.1 Simulation d'une chaîne de Markov

Dans cette partie, nous allons générer un tirage aléatoire d'une chaîne de Markov et observer l'influence des paramètres $p_{i,j}$ de la matrice $\mathscr{P}$.

1. On pose

$$\mathscr{P} = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.3 & 0.4 & 0.3 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

   Dans le notebook récupéré sur Moodle, créer la matrice de transition **P** définie ci-dessus et la liste vide $\mathbf{A} = \{\}$ qui contiendra les résultats des tirages.

   Pour générer une chaîne de Markov, nous avons besoin d'un état initial. Celui-ci va être choisi aléatoirement, de manière uniforme, dans l'ensemble $\{1,2,3\}$.

2. L'objectif de cette question est de construire une fonction **randomPFSuniform**[] qui renvoie $1, 2$ ou 3 aléatoirement de manière uniforme.
   (A) Expliquer en quoi l'algorithme 1.2 permet de simuler une loi uniforme sur $\{1,2,3\}$ à partir d'un simulateur de la loi uniforme sur $[0,1]$.
   (B) Écrire le module **randomPFSuniform**[] (vous pouvez utiliser la fonction **If**).

3. Utiliser la commande **ApendTo**[**A,randomPFSuniform**[]] pour adjoindre le résultat du premier tirage à la liste **A**.

   Étant donné **A**[[1]], le résultat de notre premier tirage, on doit choisir, pour notre second tirage, un nombre tel que la probabilité d'obtenir $i \in \{1,2,3\}$ soit $p_{\mathbf{A}[[1]],i}$.

4. En vous inspirant de la question 2 de la préparation, construire une fonction **randomPFS**[$a\_, b\_$], qui prend deux nombres $a, b \in [0,1]$ et qui renvoie 1 avec une probabilité de a, 2 avec une probabilité de b et 3 avec une probabilité de $1-a-b$. Expliquer votre algorithme.

5. Finalement, à l'aide d'une boucle, remplir la liste **A** pour qu'elle contienne un échantillon de 100 tirages aléatoires de notre chaîne de Markov.
   Notez que la commande **A**[[-1]] donne le dernier élément de la liste **A**.

6. Observer la liste obtenue. Expliquer ce que vous observez par l'analyse des coefficients de la matrice de transition $\mathscr{P}$.

```
fonction RANDOMPFSUNIFORM                                  ▷ pas d'entrée pour ce module
    n ← 0                                                                  ▷ initialisation
    x ← aléatoire uniforme sur [0, 1]                          ▷ utilisation loi uniforme
    si x ∈ [0, 1/3] alors
        n ← 1                  ▷ transformation de l'aléatoire continu en aléatoire discret
    sinon
        si x ∈ [1/3, 2/3] alors
            n ← 2
        sinon                                                  ▷ cas x ∈ [2/3, 1]
            n ← 3
        fin si
    fin si
    renvoyer n
fin fonction
```

FIGURE 1.2 – Algorithme de simulation d'une loi uniforme sur 3 entiers

1.2.2 Implémentation du jeu

Dans cette partie, nous allons en premier lieu programmer une interface pour pouvoir jouer à Pierre-Feuille-Ciseaux contre l'ordinateur qui jouera aléatoirement de manière uniforme. Puis nous allons programmer l'intelligence artificielle de l'ordinateur en utilisant la stratégie que vous avez étudiée dans la préparation.

7. Cette question fait référence au code préalablement écrit sur le notebook disponible sur Moodle.
   (A) Analyser le code du notebook et le compléter pour que la stratégie de l'ordinateur consiste à jouer aléatoirement de manière uniforme.
   (B) Expliquer ce que représente le graphique affiché en fin de partie.
8. Nous allons maintenant implémenter l'intelligence artificielle. Commencer par copier la cellule de code (pour garder une trace de la question précédente).
   (A) Créer la fonction **Jouer2**[*a_*] qui est un copier-coller de **Jouer¥**[*a_*]. De même, recopiez les parties **Initialisation** et **Affichage des résultats**.
   (B) Initialiser à zéro une matrice de taille 3×3 notée **Mem** dans la partie **Initialisation**.
   (C) Compléter la partie **Sauvegarde du coup de l'humain** de sorte que le coefficient **Mem**$[[i, j]]$ contienne le nombre de coups successifs (i, j) joués par l'humain à partir du troisième tour.
   (D) Vérifier que le code fonction toujours ! Vous pouvez afficher l'évolution dynamique de la matrice de transition avec **Dynamic**[**Mem**].
9. Il s'agit maintenant d'implémenter l'intelligence artificielle.
   (A) De même qu'à la question précédente : faire un copier-coller des cellules précédentes en créant la fonction **Jouer3**[*a_*].
   (B) Créer une liste de trois entiers aléatoires et trouvez la position du plus grand. Il est possible d'utiliser les fonctions (au choix) **Max** et **Position** ou **Ordering**.
   (C) Dans la partie **Prise de décision de l'ordinateur**, implémenter l'intelligence artificielle par la méthode développée dans les questions préparatoires.
   Indication : vous pouvez utiliser la question 6 et le fait que $p_{i,3} - p_{i,2}$ est proportionnel à $n_{i,3} - n_{i,2}$ (c'est-à-dire que vous pouvez vous contenter d'utiliser la matrice **Mem** sans avoir à créer de matrice de transition $\mathscr{P}$).
   (D) Si vous voulez afficher la matrice de transition à chaque étape et adapter votre stratégie en fonction, écrivez

```
Dynamic[Apply @@ {Divide, {Mem, Plus @@ Mem} // Transpose, {1}} // N //
MatrixForm]
```

(E) Commenter les résultats obtenus.

1.2.3 Tests statistiques

Dans cette dernière partie, nous allons vérifier, à l'aide d'un test statistique, qu'une suite de nombres générés par un être humain "au hasard" n'a pas de caractère aléatoire parfait.

10. Créer une liste **listHasard** que vous remplirez **vous-mêmes** de 100 entiers égaux à 1 ou -1 (ne pas utiliser l'aléatoire de *Mathematica*). Vous essaierez de choisir chaque composante de manière indépendante et uniformément sur $\{-1, 1\}$.

    Soit $(X_1, \ldots, X_{100})$ un vecteur aléatoire, dont les composantes X_i sont mutuellement indépendantes et uniformément distribuées sur $\{-1, 1\}$. On définit la fonction d'autocorrélation I de ce vecteur

    $$I(p) = \sum_{k=1}^{100-p} X_k X_{k+p}.$$

11. Dans cette question, on fait un peu de théorie...
    (A) Calculer la valeur de $I(0)$.
    (B) Quelle information donne $I(p)$, pour $p > 0$?
    (C) Que vaut $I(p)$ pour $p > 0$ lorsque les X_k sont des variables aléatoires mutuellement indépendantes identiquement distribuées de loi uniforme sur $\{-1, 1\}$?
12. Créer une fonction **autocorr**$[x\_, p\_]$ qui prend en arguments une liste de 100 entiers, un entier $p \in \{0, \ldots, 99\}$ et qui renvoie l'autocorrélation de la liste **x** en **p**.
13. Créer une liste **liste** composée de 100 entiers égaux à 1 ou -1 choisis de manière indépendante et uniforme sur $\{-1, 1\}$. Vous pouvez utiliser la fonction **RandomChoice**. Afficher, à l'aide de **ListPlot** l'autocorrélation de **liste** et commenter.
14. On cherche maintenant à déterminer empiriquement la loi de l'autocorrélation $I(1) = \sum_{k=1}^{99} X_k X_{k+1}$. Créer un échantillon **ech** de 5000 réalisations de la variables aléatoire $I(1)$. On pourra appliquer la méthode suivant 1.3

```
ech ← liste vide
pour i ← 1 à 5000 faire
    x ← liste de 100 entiers égaux à −1 ou 1 choisis indépendamment
    Adjoindre autocorr[x, 1] à ech                    ▷ Utiliser AppendTo
fin pour
```

FIGURE 1.3 – Méthode pour créer **ech** pour la question 14.

15. Visualiser la loi empirique de l'échantillon **ech** par exemple en utilisant la fonction **Histogram**. Calculer la moyenne et la variance empirique de **ech** [2]. En déduire une loi approximative de $I(1)$.
16. Dans cette question, on va construire un test d'hypothèses permettant de déterminer si la liste **listeHasard** de la question 10 a un caractère aléatoire parfait (composantes choisies indépendamment et uniformément sur $\{-1, 1\}$). Pour cela
    — énoncer mes hypothèses nulle et alternative,
    — utiliser la statistique $I(1)$,
    — indiquer la valeur critique du test pour un risque de première espèce $\alpha = 5\%$ et conclure.

2. Utiliser **Mean** et **Variance**.

1.2.4 S'il vous reste du temps

Il existe une version complétée de Pierre-Feuille-Ciseaux à 5 états, appelée Pierre-Feuille-Ciseaux-Lézard-Spock. Dans cette version, les ciseaux coupent le papier, le papier recouvre la pierre, la pierre écrase le lézard, le lézard empoisonne Spock, Spock casse les ciseaux, les ciseaux décapitent le lézard, le lézard mange le papier, le papier réfute Spock, Spock vaporise la pierre et la pierre écrase les ciseaux (représentés figure 1.4).

FIGURE 1.4 – Coups gagnants au jeu Pierre-Feuille-Ciseaux-Lézard-Spock

17. En vous inspirant des méthodes employées dans ce TP, créer une intelligence artificielle permettant de battre un joueur humain à Pierre-Feuille-Ciseaux-Lézard-Spock.

2

TP2 : Tests d'hypothèses

2.1 Préparation

On considère un échantillon $(X_1, \ldots, X_n)$ constitué de n variables aléatoires mutuellement indépendantes et identiquement distribuées selon une loi inconnue. Cet échantillon représente les résultats obtenus à un contrôle de synthèse de Probabilités-Statistiques ! ! Et on souhaite en réaliser une étude statistique.

1. Rechercher les définitions du coefficient d'asymétrie (aussi appelé *skewness*) et du kurtosis d'une loi de probabilité et expliquer ce qu'ils permettent de mesurer. Montrer que le coefficient d'asymétrie d'une loi normale est nul.
2. Nous souhaitons savoir si l'échantillon $(X_1, \ldots, X_n)$ est en adéquation avec une loi normale $\mathcal{N}(\mu, \sigma^2)$, et nous réaliserons pour cela un test de χ^2.
   (A) Formuler les hypothèses nulle H_0 et alternative H_1 du test de χ^2 d'adéquation à la loi $\mathcal{N}(\mu, \sigma^2)$.
   (B) Donner les estimations du maximum de vraissemblance μ' et $(\sigma')^2$ de μ et σ^2 en fonction de la réalisation $(x_1, \ldots, x_n)$ de l'échantillon $(X_1, \ldots, X_n)$, sous l'hypothèse H_0.
   (C) Soient a et b deux réels avec $a < b$. Exprimer l'effectif théorique de la classe des notes appartenant à $[a, b]$ en fonction de n, de la fonction de répartition F de la loi $\mathcal{N}(\mu', (\sigma')^2)$, de a et de b.
   (D) À quelle condition sur les effectifs des classes peut-on procéder au test de χ^2 sans regroupement de classes ?
   (E) Quelle est l'influence de la valeur de α, le risque de première espèce, sur le seuil de la zone de rejet du test ?

2.2 Travail en séance

***Mathematica* et les probabilités**

Les fonctions *Mathematica* portent des noms anglais. Afin de s'y retrouver, voici un petit lexique franco-anglais sur le thème des probabilités.

Français	**Anglais**
coefficient d'asymétrie	*skewness*
densité de probabilité	*PDF (probability density function)*
écart-type	*standard deviation*
fonction de répartition	*CDF (cumulative density function)*
loi (normale, du χ^2)	*(normal, chi square) distribution*
moyenne	*mean*
variance	*variance*

En *Mathematica*, les mêmes fonctions permettent aussi bien de calculer des paramètres théoriques que des estimations de ceux-ci, selon que l'on passe en argument une loi ou bien un échantillon.

```
In[1]:= Mean[PoissonDistribution[lambda]]
Out[1]= lambda

In[2]:= Mean[{3, 4, 5}]
Out[2]= 4
```

Attention, en *Mathematica*, on paramétrise une loi normale en donnant sa moyenne, puis son écart-type (convention différente de celle pratiquée à l'ENSEA, où le deuxième paramètre est la variance).

```
In[3]:= Variance[NormalDistribution[moy, std]]
Out[3]= std^2
```

2.2.1 Adéquation à une loi normale

Soit **data** l'ensemble des notes obtenues aux contrôles de synthèse de Probabilités et Statistiques des années 2016 et 2017, réalisation de l'échantillon $(X_1, \ldots, X_n)$. L'objectif de cette séance est de faire une étude statistique de ces résultats.

1. Donner une estimation de la moyenne (**muEstim**), de la variance (**varEstim**), du coefficient d'asymétrie (**skewnessEstim**) et du kurtosis (**kurtoEstim**) de la loi suivie par les X_i.
2. En consultant la documentation des fonctions utilisées à la question précédente, déterminer si **muEstim** et **varEstim** correspondent aux estimations trouvées à la question 2.B de la préparation.
3. Comparer les valeurs théoriques du kurtosis et coefficient d'asymétrie aux estimations trouvées à la question 1. Vous semble-t-il plausible que l'échantillon **data** soit issu d'une loi normale ?
4. Étude de la pertinence de faire un test d'adéquation à une loi normale *via* l'histogramme.
   (A) Tracer l'histogramme de **data** grâce à la fonction **Histogram**, vous utiliserez l'option **PDF** pour que le tracé soit normalisé comme une densité de probabilité et vous choisirez un nombre de classes qui soit pertinent.

(B) Superposer (vous pouvez utiliser la fonction **Show**) à l'histogramme la densité de probabilité de la loi normale de moyenne **muEstim** et de variance **varEstim**. Commenter.

5. Étude de la fonction de répartition.

   (A) Créer une fonction **fR**[$x\_$] qui renvoie la valeur de la fonction de répartition de la loi normale de moyenne **muEstim** et de variance **varEstim** en x.

   (B) Représenter **fR**[$x\_$] sur un intervalle de votre choix.

   (C) Commentez votre trcé.

   Nous souhaitons à présent tester l'adéquation de l'échantillon **data** avec une loi normale, à l'aide d'un test de χ^2 et nous devons au préalable répartir les notes dans des classes. On choisira 20 classes $]-\infty, 1[, [1, 2[, \ldots, [18, 19[, [19, +\infty[$.

6. Créer les listes **effObs** et **effTh** qui contiennent respectivement les effectifs observés et théoriques. Pour la création de **effObs**, on pourra utiliser la fonction **BinCounts** et on remarquera que toutes les notes de **data** sont dans l'intervalle $[0, 20[$. Pour la création de **effTH**, on pourra utiliser la question 2.C de la préparation et les fonction **AppendTo** et **PrependTo**. Commenter.

7. Créer une fonction **regroupeGauche**[liste$_, k_$] qui prend en entrée une liste et un entier positif k et renvoie la liste obtenue à partir de **liste** en regroupant (utiliser **Prepend**) et en sommant (utiliser **Total**) les k premiers termes ; On vérifiera que

   $$\textbf{regroupeGauche}[\{2, 4, 3, 6, 3, 7\}, 4]$$

   renvoie $\{15, 3, 7\}$.

8. À l'aide de la fonction **regroupeGauche**, effectuer un regroupement de classes de sorte à ce que tous les effectifs théoriques soient plus grands que 5 (le nombre de classes à regrouper sera déterminer "à vue de nez !"). On appellera **effThBis** et **effObsBis** les nouveaux effectifs théoriques et observés (on pourra utiliser la fonction **Total**).

9. Calculer la distance du χ^2 entre les effectifs théoriques et observés de la question précédente.

10. Conclure quant à l'adéquation de l'échantillon **data** à une loi normale pour un risque de première espèce $\alpha = 5\%$. On précisera le nombre de degrés de liberté du problème et on calculera la valeur seuil du test du χ^2 à l'aide de la fonction **InverseCDF**.

2.2.2 Prise en compte de l'asymétrie

L(histogramme de la question 4.A devrait sembler légèrement "étalé vers la droite". Dans cette partie, nous allons tester l'adéquation de **data** à une version modifiée de la loi normale, appelée loi normale asymétrique (*skew normal distribution*). Cette loi est décrite par trois paramètres (au lieu de deux pour la loi normale) : le paramètre de position $m \in \mathbb{R}$, le paramètre d'échelle $s > 0$ et le paramètre de forme $a \in \mathbb{R}$. En *Mathematica*, on représentera cette loi par l'expression symbolique **SkewNormalDistribution**$[m, s, a]$.

11. Quelle valeur de a faut-il choisir pour que **SkewNormalDistribution**$[m, s, a]$ soit une loi normale ? Il n'est pas demandé de faire des calculs ici mais plutôt de comparer les paramètres des lois concernées ou leurs densités.

12. Afin d'observer l'effet de l'asymétrie dans un cas particulier, superposer les densités de probabilité de la loi normale centrée réduite et de la loi normale asymétrique de paramètres $0, 1$ et 4.

13. Créer une fonction **logvraissemblance**[$m\_, s\_, a\_$] qui donne la Log-vraissemblance de l'échantillon **data** pour le modèle de la loi normale asymétrique (vous pouvez utiliser la fonction **Sum**).

14. Déterminer les estimations du maximum de vraissemblance **mEstim**, **sEstim** et **aEstim** des paramètres m, s, a (utiliser la fonction **FindMaximum**).

15. Supperposer l'histogramme de la question 4 avec le graphe de la densité de probabilité de la loi normale asymétrique de paramètres **mEstim**, **sEstim** et **aEstim**. Commenter.

16. Effectuer le test du χ^2 d'adéquation de l'échantillon **data** à une loi normale asymétrique et conclure.

2.2.3 Indépendance entre deux caractères

Dans cette dernière partie, nous allons tester l'indépendance (supposée) entre le sexe d'un individu et ses résultats aux contrôles de Probabilités et Statistiques. On dispose pour cela des deux sous-échantillons **dataFemmes** et **dataHommes** de notes obtenues au contrôle de synthèse de Probabilités et Statistiques de 2017.

17. Donner la table de contingence des effectifs observés.
18. Créer la table de contingence des effectifs théoriques.
19. Calculer la distance du χ^2. Pour cela, vous pouvez utiliser **Flatten** qui permet de perdre un niveau de liste (par exemple **Flatten** d'une matrice donc une liste de liste devient une simple liste).
20. Conclure quant à l'indépendance.

3

TP3 : Estimation de densité

Dans ce TP, nous utiliserons le langage Python. Il y'a plusieurs manière de l'utiliser, premièrement, il faut une partie d'écriture de programmes (au format .py) qui peut seulement être faite avec n'importe quel éditeur de texte, il y'a ensuite un logiciel de compilation des programmes Python, vous pouvez utiliser celui de votre choix. Chaque graphique demandé dans ce TP sera à enregistrer au format PiNOM1NOM2Qj.png où i est le numéro de la partie en question, j est le numéro de la question au sein de la partie considérée et NOM1 et NOM2 sont les deux noms de famille des deux membres du binôme de TP. Chaque question comportant le symbole $\star$ nécessitera la création d'un graphique à enregistrer comme spécifié ci-dessus. S'il y'a plusieurs graphiques à faire pour une seule question, ils seront nommés PiNOM1NOM2Qja.png, PiNOM1NOM2Qjb.png etc...

3.1 Préparation

Ce TP fait référence au chapitre 3 du cours qui n'a pas été traité en amphi. N'hésitez pas à vous y référer.

Considérons une réalisation $x_1, \dots, x_n$ d'un échantillon $X_1, \dots, X_n$ de variables identiques et indépendantes de densité commune f. Le but de ce TP est de comparer les noyaux utilisés pour estimer la densité commune f. Il faut bien comprendre que dans la pratique f est inconnue, ici, pour comparer l'efficacité des noyaux et la taille de la fenêtre h nous allons supposer dans une première partie que f est la densité d'une gaussienne centrée réduite, dans une seconde partie nous supposerons que f est la densité d'une loi de dimension plus grande.

1. Si $K : \mathbb{R} \to \mathbb{R}_+$ est un noyau statistique et $\mu \in \mathbb{R}$ une constante, la translation de K par la constante a, $\tau_\mu K$, est-elle encore un noyau statistique ?
2. Si $K : \mathbb{R} \to \mathbb{R}_+$ est un noyau statistique et $\lambda \in \mathbb{R}*$ une constante non nulle, montrer que $d_\lambda K$ définie pour tout $x \in \mathbb{R}$ par $d_\lambda K(x) = \frac{1}{\lambda} K\left(\frac{x}{\lambda}\right)$ est encore un noyau statistique.
3. Montrer que $K = \frac{1}{2} 1_{[-1,1]}$ est un noyau statistique, on l'appel le noyau uniforme.
4. Montrer que $K(x) = (1 - |x|) 1_{[-1,1]}(x)$ est un noyau statistique, on l'appel le noyau triangle.
5. Montrer que $K(x) = \frac{3}{4}(1-x^2) 1_{[-1,1]}(x)$ est un noyau statistique, on l'appel le noyau d'Epanechnikov.
6. Montrer que $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ est un noyau statistique, on l'appel le noyau gaussien.

Soit $h > 0$ une constante appelée la fenêtre. Soit K un noyau statistique. On considère la fonction $\widehat{f}_h$, définie pour tout $x \in \mathbb{R}$, par :

$$\widehat{f}_h(x) = \frac{1}{n} \sum_{k=1}^{n} d_h \tau_{X_i} K(x)$$

C'est l'estimation de la densité f avec la fenêtre h et le noyau K.

7. Montrer que $\widehat{f_h}$ est une densité de probabilité.

3.2 Travail en séance avec Python : partie 1

Le but de cette partie est de définir, représenter et comparer l'efficacité des quatre noyaux de la préparation pour l'estimation de la densité d'une gaussienne standard f. On suppose donc que $X_1, \ldots, X_n$ est un échantillon de taille n de variables indépendantes et identiquement distribuées selon la loi normale centré réduite de densité f. Télécharger le script TP3.py disponible sur Moodle, c'est dans ce script que vous définirez toutes les fonctions et répondrez aux questions.

1. Dans ce même script, définir quatre fonctions $K1, K2, K3, K4$ correspondant respectivement aux noyaux uniforme, triangle, d'Epanechnikov et gaussien.
2. ⋆ Représentez ces quatre noyaux sur un même graphique (utiliser une légende et des couleurs différentes). Créer une fonction pour faire cette question que vous nommerez **AllplotK** qui prendra en entrée les paramètres du graphique (le pas, xmin, xmax, les couleurs etc...) et représentera le graphique en retour.
3. Générer une réalisation de l'échantillon aléatoire X selon la loi gaussienne standard de taille n. (n est pour l'instant fixé à 100 dans le script).
4. Définir la fonction **fchapeau** qui prend comme argument une fonction K (le noyau), la fenêtre h et la réalisation de l'échantillon X et une variable x et qui retourne l'image de x par la fonction $\widehat{f_h}$.
5. ⋆ Représenter sur un même graphique la fonction f de référence ainsi que les quatres fonctions $\widehat{f_h}$ obtenues avec les noyaux $K1, K2, K3, K4$. Vous ajouterez une légende et des couleurs différentes à toutes les courbes. On fixera pour cette question $h = 2$. Vous définierez une fonction comme dans la question 2 pour faire cette question. Cette fonction sera nommée **Allplotfchapeauh2**.
6. ⋆ Refaire la question précédente avec $h = 1$. Qualitativement, est-ce que l'estimation diffère plus lorsque l'on fait varier le noyau utilisé ou la fenêtre h utilisée ? La nouvelle fonction pour cette question sera nommée **Allplotfchapeauh1**.
7. ⋆ Reprendre les deux questions précédentes pour $n = 10$ puis $n = 1000$. Pour cette question, quatre graphiques doivent être construits : le premier pour $(n, h) = (10, 2)$, le second pour $(n, h) = (10, 1)$, le suivant pour $(n, h) = (1000, 2)$ et le dernier pour $(n, h) = (1000, 1)$. Vous détaillerez votre raisonnement dans le script et commenterez les résultats obtenus. Revenir ensuite à la valeur de $n = 100$ dans le script pour la suite du TP.
8. Nous allons calculer l'erreur quadratique d'une estimation : soit

$$SCE(h) = \sum_{i=0}^{500} (\widehat{f_h}(t_i) - f(t_i))^2$$

la somme de carrés des écarts entre l'image de t_i par l'estimation $\widehat{f_h}$ et l'image de t_i par f, où $\{t_0, t_1, t_2, \ldots, t_{500}\}$ est une discrétisation de l'intervalle $[-5, 5]$ de pas $10/500$.
Autrement dit

$$-5 = t_0 < t_1 = -5 + 10/500 < t_2 = -5 + 20/500 < \cdots < t_{500} = -5 + 5000/500 = 5$$

Définir une fonction **SCE** qui prend comme paramètre une fonction (le noyau considéré), la fenêtre h, la densité de référence f et qui retourne $SCE(h)$.

9. Définir une fonction **lemeilleurh** qui prend une fonction (le noyau en question) et une autre fonction f (la référence) en paramètres et retourne l'index divisé par 100 du minimum de la liste $\{SCE(k/100)\}_{1 \leq k \leq 200}$. Pour chaque noyau, la meilleur fenêtre pour l'etimation de la fonction de référence est donnée par cette fonction.
10. ⋆ Définir alors une fonction qui représente graphiquement les quatres estimations de densité pour ces quatre noyaux avec les fenêtre obtenue via la fonction **lemeilleurh**. Définir pour ce faire, la fonction **Allplotfchapeauhoptimal**

3.3 Travail en séance avec Python : partie 2

Nous allons maintenant exploiter les fonctionnalités de **scikit-learn**. La fonction **estimationdensite** présente dans le script sert à effectuer une estimation de densité par noyau gaussien (**kernel**=$'gaussian'$) avec pour fenêtre h dont la densité de référence est un mélange gaussien (de deux gausiennes de moyennes mu1, mu2 et d'écartyypes sigma1, sigma2). Cette fonction fait appel au package scikit-learn.

1. $\star$ Executer cette fonction avec $mu1 = 0$, $mu2 = 5$ et $sigma1 = sigma2 = 1$, $N = 100$ et $h = 0.75$.
2. Comparer à tout autre paramètre fixés comme dans la question précédente, l'influence de la fenêtre h. On pourra tester des valeurs de h comprises entre 0.2 et 1.5. Commenter.
3. Faite varier les paramètres des deux lois gaussiennes qui définissent le mélange gaussien. Commenter.
4. Faite varier N et commenter.
5. $\star$ On peut aussi tester d'autres noyaux par exemple en remplaçant '*gaussian*' dans le code par '*epanechnikov*'. Réaliser ce graphique en executant la fonction **estimationdensite2** avec les mêmes paramètres que ceux de la question 1.

4 TP4 : Restauration d'image

Dans cette séance de travaux pratiques, on abordera le problème du débruitage d'images en noir et blanc par une méthode variationnelle. Du point de vue mathématique, une image est représentée par une matrice dont les coordonnées, à valeurs dans $[0, 1]$, représentent les pixels de l'image (0 pour noir, 1 pour blanc).

4.1 Préparation

4.1.1 Variation totale d'une image

Afin de simplifier les notations, on ne considérera que des images carrées de taille $n \times n$, représentées par une matrice $x = (x_{ij})_{1 \leq i,j \leq n} \in \mathrm{M}_n(\mathbb{R})$.
L'espace $\mathrm{M}_n(\mathbb{R})$ est muni de la norme euclidienne $||x|| = (\sum_{i=1}^n \sum_{j=1}^n x_{i,j}^2)^{1/2}$. On définit de plus la quantité suivante, appelée variation totale de x.

$$V(x) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \Big((x_{i,j} - x_{i+1,j})^2 + (x_{i,j} - x_{i,j+1})^2 \Big),$$

où les indice sont pris modulo n (par exemple $x_{i,n+1} = x_{i,1}$). Cette quantité mesure les variations d'intensité entre pixels voisins (deux pixels situés sur des bords opposés sont considérés comme voisins).

1. Quelles sont les images matricielles x pour lesquelles $V(x)$ est minimale ?
2. Calculer $\frac{\partial V(x)}{\partial x_{k,l}}$. On pourra remarquer que

$$V(x) = \frac{(x_{k,l} - x_{k,l-1})^2 + (x_{k,l} - x_{k-1,l})^2 + (x_{k,l} - x_{k,l+1})^2 + (x_{k,l} - x_{k+1,l})^2}{2} + \cdots \tag{4.1}$$

   où les terme en pointillés dans (4.1) ne dépendent pas de $x_{k,l}$.
   La variation totale d'un bruit généré aléatoirement, comme à la figure (4.1), est typiquement élevée, deux pixels voisins pouvant prendre des valeurs éloignées. C'est ce que l'on va vérifier à la question suivante.
3. On considère une image aléatoire $X = (X_{i,j})_{1 \leq i,j \leq n}$ où les variables $X_{i,j}$ sont indépendantes et identiquement distribuées selon la loi $\mathcal{U}([0, 1])$. Montrer que

$$E(V(X)) = \frac{n^2}{6}$$

4. Pour $n = 200$, l'ordre de grandeur de la variation totale d'une image "de la vie réelle" est aux alentours de 1000. Comparer cette valeur à la variation totale d'un bruit de type de la figure (4.1).

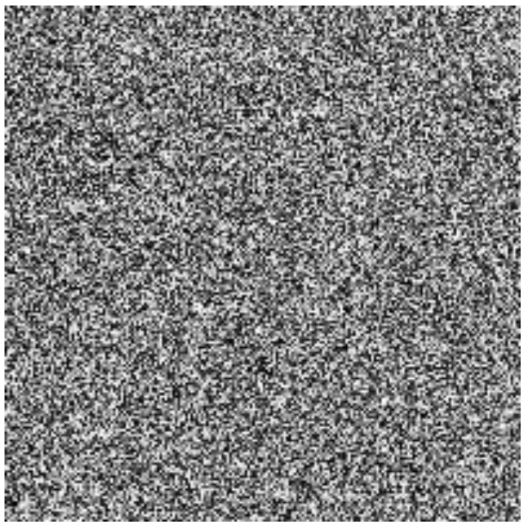

FIGURE 4.1 – Tirage d'une image aléatoire X, où les pixels $X_{i,j}$ sont indépendants et identiquement distribués selon la loi $\mathcal{U}([0,1])$.

4.1.2 Problème d'optimisation

On fixe maintenant une image bruitée a_0, dont la variation totale prend une valeur intermédiaire entre celle d'une image non bruitée (≈ 1000) et celle d'un bruit pur du type de la figure (4.1). Pour débruiter a_0, on va résoudre un problème d'optimisation. On cherche à minimiser la fonction

$$\begin{aligned} f\colon \mathrm{n}(\mathbb{R}) &\to \mathbb{R} \\ x &\mapsto V(x) + \frac{\lambda}{2}||x - a_0||^2, \end{aligned}$$

où $\lambda > 0$ est un paramètre. L'image x^* en laquelle ce minimum est atteint étant l'image restaurée.

5. Interpréter le terme $\frac{\lambda}{2}||x - a_0||^2$ dans $f(x)$. Quel est le rôle du paramètre λ ?
6. Calculer $\frac{\partial f(x)}{\partial x_{k,l}}$.

4.2 Travail en séance de TP avec Mathematica

4.2.1 Calcul de variations totales

Remarquons que la variation totale de $x \in \mathrm{M}_n(\mathbb{R})$ est donnée par :

$$V(x) = \frac{||x - x'||^2 + ||x - x''||^2}{2},$$

où x' (respectivement x'') est obtenue à partir de x en permutant les lignes (respectivement les colonnes) de manière circulaire vers le haut (respectivement vers la gauche).

1. En utilisant l'aide de la fonction **RotateLeft**, trouver comment obtenir x' et x'' à partir de x. Tester sur l'exemple

$$x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

2. Créer une fonction **variationTotale**[x_] qui prend en entrée une image matricielle x et renvoie sa variation totale. La norme euclidienne d'une matrice x est donnée par **Norm**[x, "Frobenius"].
On définit une constante globale $n = 200$ (ne pas changer cette constante, il s'agit de la taille de l'image de la question 4).

3. Dans cette question, on étudie le cas particulier d'une image crée par loi uniforme.
   (A) Créer une image matricielle de taille $n \times n$ dont les coordonnées sont des réalisations indépendantes de la loi $\mathcal{U}([0,1])$ (utiliser la fonction **RandomReal**).
   (B) Visualiser cette image à l'aide de **Image**.
   (C) Calculer la variation totale de cette image. Commenter.
4. L'objectif de cette question est d'étudier l'image figurative de taille 200×200.
   (A) Convertir l'image donnée dans le notebook en une matrice **mat** à l'aide de **ImageData**.
   (B) Calculer sa variation totale. Commenter.
5. Dans cette question, on bruite l'image figurative étudiée précédemment et on étudie cette nouvelle image bruitée.
   (A) Créer la matrice a_0 en rajoutant à **mat** un bruit blanc gaussien d'écart-type $\sigma = 0,2$. (Utiliser **RandomVariate** et **NormalDistribution**).
   (B) Visualiser a_0.
   (C) Calculer sa variation totale. Commenter.

4.2.2 Minimisation par méthode du gradient

Dans cette partie, nous allons minimiser la fonction f de la partie 4.1.2 de la préparation en utilisant l'algorithme de descente par gradient à pas constant, en partant de l'image initiale a_0 de la question 5. Le paramètre λ ainsi que le pas de descente μ sont des constantes globales définies dans le notebook.
On rappelle que la méthode de descente par gradient est un algorithme itératif, on notera $a_0, a_1, a_2, \ldots$ les étapes intermédiaires de l'algorithme.

6. Pourquoi minimise-t-on f en partant de l'image initiale a_0 ?
7. Créer une fonction **nextStep**[x_] qui prend en argument une matrice x et renvoie l'itérée suivante (ainsi, si l'argument d'entrée est a_k, la valeur retournée est a_{k+1}).
8. Créer la liste **pointsIntermediaires** égale à $\{a_0, a_1, \ldots, a_{60}\}$. Utiliser la fonction **NestList**.
9. À l'aide de la fonction **Manipulate**, visualiser les matrices a_k, en faisant varier k de 0 à 60. Commenter.
10. Afficher la variation totale de a_k en fonction de k, pour k allant de 0 à 60.
11. Calculer l'erreur commise à chaque itération.
12. Jouer avec les paramètres μ et λ. Commenter.

4.2.3 Signal audio (s'il vous reste du temps)

Nous allons appliquer la même méthode de restauration à un signal audio, qui est unidimensionnel. L'espace euclidien des signaux considérés est tout simplement $\mathbb{R}^n$.

13. Restaurer le signal audio donné dans le notebook, en appliquant une méthode analogue et en respectant les points suivants.
    — La variation totale d'un signal $x \in \mathbb{R}^n$ est cette fois définie par
    $$V(x) = \sum_{k=1}^{n} (x_k - x_{k+1})^2,$$
    où $x_{n+1} = x_1$ dans cette notation.
    — On ne mémorisera pas la liste de tous les points intermédiaires comme cela a été fait précédemment.
    — Le nombre d'itérations dans la méthode du gradient ne doit pas être fixé a priori comme précédemment. À la place, vous utiliserez une condition d'arrêt.
14. Tester votre algorithme de restauration avec différentes valeurs du paramètre λ et commenter les résultats obtenus.