Modèle de rupture pour la loi de Weibull sur des données du CAC 40

Groupe 1

Ikrame BOULIF, Mohammed Amine BOUKIR, Souhayla HEDRI, Qian LIU, Bealy MECH, Arian NAJAFY ABRANDABADY, Nada NASLOUBY, Adrien NAVARRO, Abdelouahab ZAARI

18 Avril 2022

Séries Temporelles - Modélisation statistique Master 1 - Mathématiques Appliquées

Table des matières

Résumé

Dans cette étude, nous nous intéressons au comportement d'un modèle logarithmique pour la loi de Weibull appliqué sur des données financières de l'indice français du CAC 40. Tout d'abord, notre but est de détecter le point pour lequel le modèle tend à changer sa loi de distribution. On appelle ce point le point de rupture. Puis, nous réalisons un test de Kolmogorov-Smirnov pour vérifier la pertinence du modèle établi sur nos données, c'est-à-dire voir s'il existe une rupture significative dans la loi du modèle ou non. Pour ce faire, nous proposons deux méthodes, la première repose sur la maximisation du log vraisemblance et la deuxième est une méthode analytique qui consiste à linéariser la fonction de répartition de la loi de Weibull.

Mots-clés : Loi de Weibull, Détection de la rupture, point de rupture, Modèle l'algorithmique, Test de Kolmogorov-Smirnov .

1 Introduction

Les modèles de rupture ont été étudiés pour identifier un changement dans la probabilité de distribution d'un processus stochastique ou d'une série temporelle. Autrement dit, pour une séquence donnée de variables aléatoires $(X_i)_{0\leq i\leq n}$, nous essayons de trouver un point de rupture k où les éléments $X_1, ..., X_k$ ont une fonction de distribution identique f_1 et les éléments $X_{k+1}, ..., X_n$ sont distribuées selon une autre densité de probabilité f_2.

Dans les données du CAC 40, ils existent plusieurs entreprises dont la valeur ajustée à la clôture X_t de ses actions contribue dans le calcul de cet indice. Notamment, nous voulons étudier le comportement de :

$$Y_t = \log\left(1 + \frac{X_t}{X_{t-1}}\right) \tag{1}$$

Soit alors n le nombre des observations X_i de chaque entreprise. La relation entre X et Y est décrite par l'équation (1), où la distribution de Y change de structure après un point de rupture $k \in \{4, ..., n-4\}$. Cette restriction sur k est nécessaire pour s'assurer que les paramètres du modèle sont estimables. Le modèle de rupture est donné alors par :

$$\begin{cases} Y_t \sim W(a_1, b_1) & t \leq k \\ Y_t \sim W(a_2, b_2) & t > k \end{cases} \tag{2}$$

tels que : a_1, b_1, a_2, b_2 sont les paramètres du modèle de rupture de la loi de Weibull, et k et le point de changement.

2 Statistiques

2.1 Loi de Weibull

La loi Weibull à deux paramètres est une loi de probabilité qui a été largement utilisée comme modèle probabiliste dans des études sur les temps de survie, on peut définir sa densité comme suit :

$$f(x,a,b) = \frac{b\,x^{b-1}}{a^b}\exp\Big(-\Big(\frac{x}{a}\Big)^b\Big),\ \ x>0,\ \ a>0,\ \ b>0$$

On sait que chaque densité d'une loi est définie par une fonction de répartition, c'est la cas pour le modèle de Weibull, sa fonction de répartition est définie comme suit :

$$F(x,a,b) = 1 - \exp\Big(-\Big(\frac{x}{a}\Big)^b\Big)$$

On peut concevoir que la fonction de répartition dépend aussi des paramètres a, b de la fonction densité de Weibull. Ainsi pour avoir une bonne illustration de cette loi, qui nous sera utile dans la partie de modélisation, on fait varier le paramètre (a) et on garde le deuxième paramètre (b) constant dans le graphe de la fonction de densité ci-dessous :

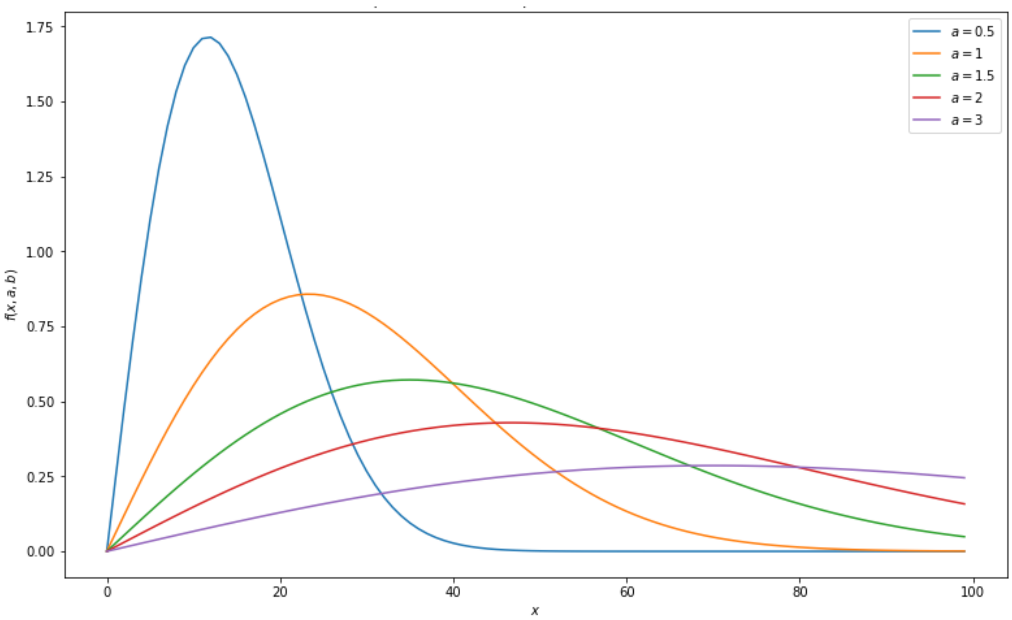

FIGURE 1 – Les densité de probabilité de la loi de Weibull pour différentes valeurs de a et $b = 2$

On peut bien analyser d'après la figure 1 que plus on augmente la valeur a de la fonction densité de Weibull plus la variance de la densité augmente ainsi pour $a > 1$, la densité de la loi de Weibull tend vers zéro à mesure que x augmente. Dans ce cas, elle n'admet qu'un seul mode qui tend rapidement vers b lorsque a tend vers l'infini.

Pour $0 < a < 1$, le mode est zéro et la densité est une fonction décroissante de x en tout point du domaine, donc finalement on converge vers une densité constante, alors on constate clairement que les paramètres de la fonction jouent un rôle sur la fonction densité du modèle, donc c'est la raison pour laquelle dans les parties suivantes on va s'intéresser à estimer ces paramètres par des méthodes statistiques d'estimations qui sont bien définies.

2.2 Méthode du maximum de vraisemblance

(cf. partie 2 de l'article [1])

La première méthode pour notre modèle (2) que nous proposons pour estimer les paramètres de la loi de Weibull, repose sur le maximum de vraisemblance.

Soit $\underline{x} = (x_1, ..., x_n)$

La fonction de vraisemblance s'écrit :

$$L(\underline{x}, \theta) = \prod_{i=1}^{k} f(x_i, \theta) = \prod_{i=1}^{k} f_1(x_i, \theta). \prod_{i=k+1}^{n} f_2(x_i, \theta) \tag{3}$$

On définit alors la log vraisemblance par :

$$l(\underline{x}, \theta) = \log\left(L(x, \theta)\right) = \sum_{i=1}^{k} \log\left(f_1(x_i, \theta)\right) + \sum_{i=k+1}^{n} \log\left(f_2(x_i, \theta)\right) \tag{4}$$

telles que f_1 et f_2 sont les fonctions densité de probabilités des lois $W(a_1, b_1)$ et $W(a_2, b_2)$ respectivement :

$$f_1(x_i, \theta) = \frac{b_1 x_i^{b_1-1}}{b_1} \exp\left(-(\frac{x_i}{a_1})^{b_1}\right)$$

et

$$f_2(x_i, \theta) = \frac{b_2 x_i^{b_2-1}}{b_2} \exp\left(-(\frac{x_i}{a_2})^{b_2}\right)$$

On a donc :

$$-l(\underline{x}, \theta) = \sum_{i=1}^{k} \left(\frac{x_i}{a_1}\right)^{b_1} + \sum_{i=k+1}^{n} \left(\frac{x_i}{a_2}\right)^{b_2} - \sum_{i=1}^{k} \log\left(\frac{b_1 (x_i)^{b_1-1}}{a_1^{b_1}}\right) - \sum_{i=k+1}^{n} \log\left(\frac{b_2 (x_i)^{b_2-1}}{a_2^{b_2}}\right) \tag{5}$$

On cherche alors à estimer le vecteur des paramètres $\theta = (a_1, b_1, a_2, b_2, \text{k})$ par la méthode du maximum de vraisemblance.

L'idée est que pour chaque $k \in \{4, ..., n-4\}$, on maximise le log vraisemblance correspondante. On récupère alors le vecteur $(a_1^*, b_1^*, a_2^*, b_2^*)$ qui est le vecteur des estimateurs des paramètres de la loi de Weibull.

L'estimateur du point de rupture k s'obtient donc par :

$$k^* = \underset{k}{argmax}\; l\left(\underline{x}, (a_1^*, b_1^*, a_2^*, b_2^*, k)\right) \tag{6}$$

k_0	échantillon$_1(k_0)$	échantillon$_2(k_0)$
4	$X_1, ..., X_4$	$X_5, ..., X_n$
5	$X_1, ..., X_5$	$X_6, ..., X_n$
$\vdots$	$\vdots$	$\vdots$
n-4	$X_1, ..., X_{n-4}$	$X_{n-3}, ..., X_n$

Table 1 – Répartition des données en deux échantillons

Vu que la résolution théorique de ce problème d'optimisation n'aboutit pas à des expressions analytiques des estimateurs, on a recours à la fonction "minimize" du module `scipy` du langage Python. Nous allons chercher à minimiser la quantité (5).

On a défini tout d'abord une fonction "log_vraisemblance_weibull" qui calcule la log vraisemblance de la loi de Weibull de paramètres (a_1, b_1) puis on a défini une autre fonction "log_vraisemblance" qui prend en paramètre la rupture et applique la log vraisemblance précédente sur deux lois de Weibull de paramètres (a_1, b_1, a_2, b_2). Et enfin on a défini la fonction "estimation" qui estime le moment de rupture ainsi que les quatre paramètres en fonction des données.

Le code Python est en annexe.

2.3 Méthode analytique

2.3.1 Estimation des paramètres du modèle linéaire simple

(cf partie 4 de l'article [1])

L'objectif dans cette méthode est d'étudier un modèle de rupture linéaire et puis de l'appliquer sur la loi de Weibull.

On considère un modèle de rupture pour une régression linéaire simple avec un seul point de changement.

Soit n couples d'observations (X_i, Y_i) telle que la relation entre X et Y est décrite par une régression linéaire simple et que la structure change après un certain point $k \in \{4, ..., n-4\}$.

Donc, les observations (X_i, Y_i) suivent un certain modèle linéaire pour $i \leq k$ et un autre modèle linéaire pour $i > k$.

Le modèle de rupture est donné alors par :

$$\begin{cases} Y_i = B_1 + A_1 h(X_i) + \epsilon_{1,i} & i = 1, ..., k \\ Y_i = B_2 + A_2 h(X_i) + \epsilon_{1,i} & i = k+1, ..., n \end{cases} \tag{7}$$

où A_j et B_j (j = 1, 2), sont les inconnues du modèle de rupture, les $\epsilon_{i,j}$ sont des erreurs indépendantes et h est une fonction connue.

Pour k_0 dans $\{4, ..., n-4\}$, on construit $n-7$ deux échantillons comme suit :

Pour chaque $k_0 \in \{4, ..., n-4\}$, on considère les modèles suivants :

$$\begin{cases} Y_i = B_1(k_0) + A_1(k_0)h(X_i) + \epsilon_{1,i}(k_0) & i = 1, ..., k \\ Y_i = B_2(k_0) + A_2(k_0)h(X_i) + \epsilon_{2,i}(k_0) & i = k+1, ..., n \end{cases} \tag{8}$$

Pour chaque $k_0 \in \{4, ..., n-4\}$, $A_1(k_0), B_1(k_0), A_2(k_0)$ et $B_2(k_0)$, on résout le problème de minimisation :

$$\underset{(A_1(k_0),B_1(k_0),A_2(k_0),B_2(k_0))}{min} D(k_0, A_1(k_0), B_1(k_0), A_2(k_0), B_2(k_0))$$

où :

$$D(k_0, A_1(k_0), B_1(k_0), A_2(k_0), B_2(k_0)) = \sum_{i=1}^{k_0} \epsilon_{1,i}(k_0)^2 + \sum_{i=k_0+1}^{n} \epsilon_{2,i}(k_0)^2 \tag{9}$$

Tous calculs faits, on obtient les estimateurs de $A_1(k_0), B_1(k_0), A_2(k_0)$ et $B_2(k_0)$:

$$\begin{cases} \hat{A}_1(k_0) = \dfrac{\sum_{i=1}^{k_0} \left(h(X_i) - \bar{h}_{k_0}(X)\right)(Y_i - \bar{Y}_{k_0})}{\sum_{i=1}^{k_0} (h(X_i) - \bar{h}_{k_0}(X))^2}, & \hat{B}_1(k_0) = \bar{Y}_{k_0} - \hat{A}_1(k_0)\bar{h}_{k_0}(X) \\ \hat{A}_2(k_0) = \dfrac{\sum_{i=1}^{k_0} (h(X_i) - \bar{h^*}_{k_0}(X))(Y_i - \bar{Y}_{k_0})}{\sum_{i=1}^{k_0} (h(X_i) - \bar{h^*}_{k_0}(X))^2}, & \hat{B}_2(k_0) = \bar{Y^*}_{k_0} - \hat{A}_2(k_0)\bar{h^*}_{k_0}(X) \end{cases} \tag{10}$$

où :

$$\begin{cases} \bar{h}_{k_0}(X) = \dfrac{1}{k_0} \sum_{i=1}^{k_0} h(X_i), & \bar{h^*}_{k_0}(X) = \dfrac{1}{n-k_0} \sum_{i=k_0+1}^{n} h(X_i) \\ \\ \bar{Y}_{k_0} = \dfrac{1}{k_0} \sum_{i=1}^{k_0} Y_i, & \bar{Y^*}_{k_0} = \dfrac{1}{n-k_0} \sum_{i=k_0+1}^{n} Y_i \end{cases} \tag{11}$$

Posons $D(k_0) = D(k_0, \hat{A}_1(k_0), \hat{B}_1(k_0), \hat{A}_2(k_0), \hat{B}_2(k_0))$ et

$$k^* = \underset{k_0}{argmin}\ D(k_0) \tag{12}$$

De l'équation (10), on déduit les estimateurs de $A_1(k_0), B_1(k_0), A_2(k_0)$ et $B_2(k_0)$:

$$\hat{A}_1 = \hat{A}_1(k^*), \hat{B}_1 = \hat{B}_1(k^*), \hat{A}_2 = \hat{A}_2(k^*), \hat{B}_2 = \hat{B}_2(k^*) \tag{13}$$

et le point de changement est estimé alors par :

$$\hat{k} = taille\ de\ echantillon_1(k^*) \tag{14}$$

2.3.2 Estimation des paramètres de la loi de Weibull

Dans cette partie, notre objectif est d'estimer les paramètres (a, b) de la distribution de Weibull présentée dans la partie 2.1 et qu'on note $W(a, b)$.

Nous assumons avoir une séquence d'observations $Y_1, ..., Y_n$ dont la distribution représente une rupture selon le modèle (2).

k_0	échantillon$_1$(k_0 ordonné)	échantillon$_2$(k_0 ordonné)
4	$X_1^{(1)}, ..., X_1(4)$	$X_2^{(1)}, ..., X_2^{(n-4)}$
5	$X_1^{(1)}, ..., X_1^{(5)}$	$X_2^{(1)}, ..., X_2^{(n-5)}$
$\vdots$	$\vdots$	$\vdots$
n-4	$X_1^{(1)}, ..., X_1^{(n-4)}$	$X_2^{(1)}, ..., X_2^{(4)}$

TABLE 2 – Répartition des données en deux échantillons ordonnés. $X_j^{(i)}$ représente le i-ème ordre statistique de l'$échantillon_1(k_0)$ (j = 1,2).

Pour chaque $k_0 \in \{4, ..., n-4\}$, on construit $n-7$ deux échantillons et on les ordonne comme représenté sur la Table 2.

On introduit deux grandeurs qui sont les approximations de **Bernard** & **Bosi-Levenbach** [5] pour les rangs médians :

$$\begin{cases} MR_1(i) = \frac{i-0.3}{k_0+0.4} & \text{pour } i \in 1, \dots, k \\ MR_2(i) = \frac{i-0.3}{n-k_0+0.4} & \text{pour } i \in k_0+1, \dots, n \end{cases} \tag{15}$$

La fonction de répartition de la loi de Weibull est alors transformée en une fonction linéaire :

$$ln(-ln(1-F(x))) = bln(x) - bln(a)$$

Posons Y = ln(-ln(1 - F(x))), A = b et B = -b ln(a).

Les rangs médians (15) vont nous permettre de déterminer le modèle de rupture :

$$\begin{cases} Y_i = \ln(-\ln(1-MR_1(i))) & \text{pour } i \in 1, \dots, k_0 \\ Y_i = \ln(-\ln(1-MR_1(i))) & \text{pour } i \in k_0+1, \dots, n \end{cases} \tag{16}$$

On va ensuite réaliser une régression linéaire pour déterminer les coefficients a_1, b_1, a_2, b_2. En effet, on peut écrire les Y_i' de la manière suivante :

$$\begin{cases} Y_i = B_1(k_0) + A_1(k_0)ln(X_i) & \text{pour } i \in 1, \dots, k_0 \\ Y_i = B_2(k_0) + A_2(k_0)ln(X_i) & \text{pour } i \in k_0+1, \dots, n \end{cases} \tag{17}$$

avec k_0 le point de rupture.

D'après les équations (12), (13) et (14), on déduit les estimateurs des paramètres de la loi de Weibull a_1, b_1, a_2 et b_2 :

$$\begin{cases} \hat{a}_j = \exp\left(-\frac{\hat{B}_j(k^*)}{\hat{b}_j}\right), \hat{b}_j = \hat{A}_j(k^*), \qquad j = 1,2 \end{cases} \tag{18}$$

2.4 Tests de Kolmogorov Smirnov d'homogénéité

2.4.1 Présentation théorique du test

(cf chapitre 4 du document [3])

Le test de Kolmogorov-Smirnov (K-S) est un test statistique standard pour décider si un ensemble de données est cohérent avec une distribution de probabilité donnée. Le test classique est unidimensionnel - chaque point de données est un nombre unique -.

Mais il existerait également un test KS à deux échantillons similaire, qui comporte deux ensembles de données au lieu d'un ensemble de données et d'un modèle (Fig.2). Celui-ci vérifie si deux ensembles de données sont tous deux tirés de la même distribution de probabilité sous-jacente, mais sans supposer de modèle spécifique pour cette distribution.

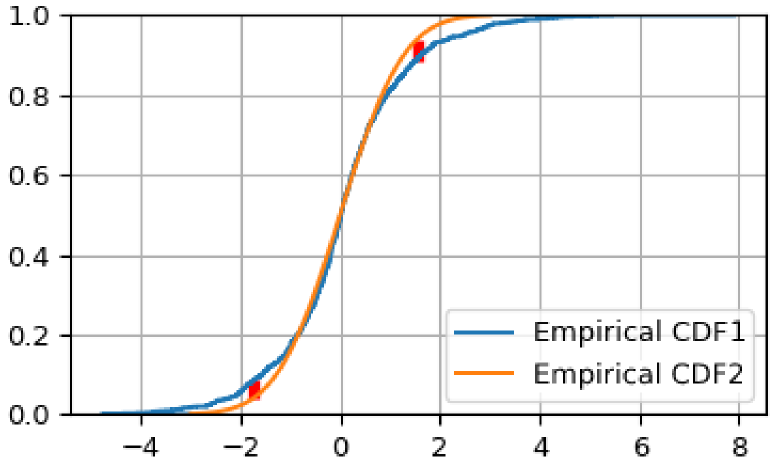

Figure 2 – Exemple du test de K-S d'homogénéité. Les lignes rouges sont les distances maximum des deux fonctions de répartition empiriques.

En pratique, le test K-S serait effectué avec un plus grand nombre de points de données dans chaque ensemble, pas moins d'ordre 10 ou 20.

Nous tenterons dans cette partie de présenter le test de Kolmogorov Smirnov d'homogénéité.

Nous considérons tout d'abord un n-echantillon $(X_1, ..., X_n)$ d'une variable aléatoire X et sa fonction de répartition qu'on notera F.

Nous avons donc :

$$\forall t \in \mathbb{R}, F(t) = \mathbb{P}(X \leq t) = \mathbb{P}(X_i \leq t)$$

Le but serait de tenter d'estimer F en introduisant la fonction de répartition empirique. Celle-ci associé à l'échantillon que nous venons de décrire s'écrit sous la forme :

$$\mathbb{R} \to [0, 1]$$
$$t \mapsto F_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq t\}}$$

La fonction F_n est une fonction en escalier, croissante, continue à droite et admettant une limite à gauche. Elle est également discontinue aux points $(X_i)_1$ et constante sur $[X_i, X_{i+1}]$ pour $i \in 1, ..., n-1$. Elle est un estimateur naturel sans biais et fortement consistent de $F(t)$. Par ailleurs :

$$\forall t \in \mathbb{R}, \sqrt{n}(F_n(t) - F(t)) \xrightarrow[n \to +\infty]{} \mathcal{N}(0, F(t)(1 - F(t))$$

Nous allons maintenant introduire le théorème de Glivenko-Cantelli.

En considérant une suite de variables aléatoires $(X_i)_{i \geq 1}$ i.i.d. de fonction de répartition F, on pose :

$$\forall t \in \mathbb{R}, F_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq t\}}$$

Alors on a :

$$\sup_{t \in \mathbb{R}} |F_n(t) - F(t)| \to 0 \quad p.s.$$

Le théorème de Glivenko-Cantelli est une généralisation de la loi forte des grands nombres au cas non-paramétrique.

Nous donnons la généralisation du TCL dans ce qui suit. Tout en considérant un n-echantillon issu de X et en notant F la fonction de répartition de X ? et F_n la fonction de répartition empirique. Si F est continue, la loi de :

$$D(F_n, F) = \sup_{t \in \mathbb{R}} |F_n(t) - F(t)|$$

ne dépend pas de F.

La statistique introduite dans ce théorème nous permettra de construire un test d'ajustement à une loi (test de Kolmogorov). Dans le même esprit, nous arriverons aussi à construire un test d'homogénéité (test de Kolmogorov - Smirnov).

En nous basant sur la proposition précédente et en considérant les mêmes hypothèses que ci-dessus, nous pouvons déduire le théorème qui suit. Tel que tout en supposant que la variable aléatoire $\sqrt{n}D(F_n, F)$ converge en loi, vers une loi limite qui ne dépend pas de F et dont la fonction de répartition s'écrira sous la forme :

$$\forall t \geq 0, F_{KS}(t) = 1 + 2 \sum_{k=1}^{+\infty} (-1)^k exp(-2k^2 t^2)$$

On a donc, pour $\lambda > 0$,

$$\mathbb{P}(\sqrt{n}D(F_n, F) \leq \lambda) \to 1 + 2 \sum_{k=1}^{+\infty} (-1)^k exp(-2k^2 \lambda^2)$$

Ce théorème est admis.

Nous pouvons donc construire un test d'ajustement à une loi, dit test de Kolmogorov. On peut d'abord remarquer que si on réordonne de manière croissante l'échantillon, $X_1, ..., X_n$ alors $F_n(X_j) = \frac{j}{n}$ et :

$$D(F_n, F) = \max_{1\leq j\leq n} max(|\frac{j}{n} - F(X_{(j)})|, |F(X_{(j)}) - \frac{j-1}{n}|)$$

Si on veut tester que la loi de l'échantillon a pour fonction de répartition F_0, c'est-à-dire $H_0 : F = F_0$ contre $H_1 : F = F_1$, on commence par réordonner l'échantillon. Puis on calcule $D(F_n, F)$, en remarquant que sous H_0, on a $D(F_n, F) = D(F_n, F_0)$. Puis on cherche (dans une table) le quantile $k_{1-\alpha}$ de la loi de Kolmogorov. On accepte alors H_0 si $D(F_n, F_0) < D(F_n, F_0))$. Ce test est asymptotiquement de niveau α et sa puissance tend vers 1 quand n tend vers $+\infty$.

Et dans le même esprit alors, nous allons construire un test d'homogénéité.

On observe deux échantillons de taille respective n et m, $(X_1, ...X_n)$ et $(Y_1, ...Y_m)$. On veut tester si les deux échantillons sont issus d'une même loi (éventuellement inconnue). On note F la fonction de répartition de chacune des variables X_i et G la fonction de répartition de chacune des variables Y_i. On veut tester $H_0 : F = G$ contre $H_1 : F \neq G$. Pour cela, on introduit :

$$\forall t \in \mathbb{R}, F_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{X_i\leq t\}}$$

et :

$$G_m(t) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{Y_i\leq t\}}$$

et on pose :

$$D_{m,n} = \sup_{t\in\mathbb{R}} |F_n(t) - G_m(t)|$$

Par conséquent, avec les hypothèses ci-dessus et sous "$H_0 : F = G$", on a le résultat du théorème suivant :

$$\mathbb{P}(\sqrt{\frac{nm}{n+m}}D_{m,n} \leq \lambda) \rightarrow 1 + 2\sum_{k=1}^{+\infty}(-1)^k e^{-(2k^2\lambda^2)}$$

Ainsi ceci permet de construire un test sur le même modèle que ci-dessus.

2.4.2 Cas pratique du test

Notons les données brutes X_1 X_2 ... X_n et Y_1 Y_2 ... Y_m, séparées par la rupture à X_n .

Notons les données ordonnées $X_{(1)}$ $X_{(2)}$ ... $X_{(n)}$ et $Y_{(1)}$ $Y_{(2)}$ ... $Y_{(m)}$ pour X, Y

Les données ordonnés peuvent être écrites comme :

$$X_{(1)}X_{(2)}Y_{(1)}X_{(3)}X_{(4)}Y_{(2)}...X_{(n)}Y_{(m-1)}Y_{(m)},$$

et on note leur valeurs comme :

$$t_1\ t_2\ t_3\ t_4\ t_5\ t_6 \quad ... \quad t_{n+m-1}\ t_{m+n}.$$

Pour chaque t_i, on peut calculer $|F_n(t_i) - G_n(t_i)|$ avec

$$F_n(t_i) = P(X \leq t_i) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq t_i\}}$$

et

$$G_n(t_i) = P(Y \leq t_i) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{Y_i \leq t_i\}}$$

On a donc

$$D(t_i) = |F_n(t_i) - G_n(t_i)| = \left| \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq t_i\}} - \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{Y_i \leq t_i\}} \right|.$$

Par exemple ici,

$$D(t_1) = |\frac{1}{n} - 0|$$
$$D(t_2) = |\frac{2}{n} - 0|$$
$$D(t_3) = |\frac{2}{n} - \frac{1}{m}|$$
$$...$$

En effet, on a

$$D_{m,n}^{obs} = \sup_{t \in \mathbb{R}} |F_n(t) - G_n(t)|$$

En posant $\lambda = \sqrt{\frac{nm}{n+m}} D_{m,n}^{obs}$, on a donc

$$P(\sqrt{\frac{nm}{n+m}} D_{m,n} \leq \lambda) \longrightarrow 1 + 2\sum_{k=1}^{+\infty} (-1)^k e^{-(2k^2\lambda^2)}$$

Donc,

$$\text{p-value} = P(\sqrt{\frac{nm}{n+m}} D_{m,n} \geq \lambda) \longrightarrow -2\sum_{k=1}^{+\infty} (-1)^k e^{-(2k^2\lambda^2)}$$

Pour une précision de ϵ de la p-value, on somme jusqu'à N tel que $e^{-2N^2\lambda^2} < \epsilon$.

C'est à dire

$$N \geq \sqrt{\frac{log\epsilon}{-2\lambda^2}}$$

Cette approximation fonctionne pour n et m grands. En général, on peut l'utiliser dès $n > 20$ et $m > 20$.

Pour n et m petits, il faut connaître la loi exacte, avec la table de la loi.

Après avoir comparé avec les résultats générés par la commande `scipy.stats.ks_2samp` de python, on confirme que notre algorithme marche bien. Les résultats des deux algorithmes ont seulement moins de 10^{-3} d'écart. On en conclut ainsi que python a utilisé autre méthode que ce qu'on utilise pour réaliser ce test.

3 Application sur les données CAC 40

Dans cette section, on applique les deux méthodes présentées précédemment sur les données du CAC 40 que nous avions extraites à partir du 1er janvier 2019.

3.1 Résultats

3.1.1 Méthode théorique

Tous ces k^* ont passé le test de K-S , avec des p-values toutes plus petites que 10^{-3}, et la majorité sont même plus petites que 10^{-10}.

	AC.PA	ACA.PA	AI.PA	AIR.PA	ALO.PA	BN.PA	BNP.PA	CA.PA
a_1	0.6964	0.6976	0.6961	0.6975	0.6971	0.6955	0.6974	0.6967
b_1	93.2585	88.2941	150.1628	93.7498	93.8283	174.1513	90.7773	82.8035
a_2	0.7010	0.6997	0.6969	0.7021	0.6986	0.6969	0.7007	0.6984
b_2	31.6030	45.3923	81.4529	29.5291	48.6494	74.4169	36.6760	47.9260
k^*	300	292	292	292	287	292	292	296

	CAP.PA	COFA.PA	CS.PA	DG.PA	DGM.PA	EDF.PA	EN.PA	GLE.PA
a_1	0.6996	0.6977	0.6960	0.6964	0.7104	0.6963	0.6997	0.6977
b_1	45.9255	67.6736	134.9967	129.9169	12.8679	89.5455	41.9164	77.4924
a_2	0.6978	0.7008	0.6993	0.6997	0.6998	0.6997	0.6961	0.7027
b_2	82.8096	39.6450	39.5671	36.6917	44.3831	45.3980	120.0483	25.1274
k^*	403	302	290	292	329	266	476	295

	HCMLF.PA	HO.PA	IDL.PA	IPN.PA	KER.PA	LR.PA	MC.PA	ML.PA
a_1	0.7000	0.6987	0.7117	0.7007	0.6992	0.6988	0.6987	0.6990
b_1	40.1438	45.8628	43.8972	33.3433	50.5596	41.1309	61.1711	47.2774
a_2	0.6978	0.6963	0.6989	0.6970	0.6978	0.6966	0.6974	0.6974
b_2	70.3117	112.2680	58.3544	90.0097	78.7903	115.5117	95.8124	94.5262
k^*	357	480	5	472	368	365	478	368

	OR.PA	ORA.PA	PUB.PA	RI.PA	RNO.PA	SAF.PA	SAN.PA	SEV.PA
a_1	0.6962	0.6952	0.6963	0.6970	0.7018	0.6969	0.6971	0.6989
b_1	130.9861	160.2303	106.6904	78.8740	31.6511	93.3112	82.9666	40.3092
a_2	0.6976	0.6969	0.6998	0.6961	0.6985	0.7018	0.6958	0.6936
b_2	69.8910	60.1818	43.9953	126.9085	63.9343	29.2425	137.8265	693.6804
k^*	211	285	292	475	486	300	477	581

	SGO.PA	SU.PA	TTE.PA	VIE.PA	VIV.PA	VK.PA
a_1	0.6994	0.6989	0.6957	0.6963	0.6972	0.7038
b_1	48.4942	52.3415	134.6228	148.9294	74.9619	22.8899
a_2	0.6977	0.6973	0.6996	0.6985	0.6986	0.7017
b_2	88.1965	96.9727	40.1571	58.4583	32.8267	37.9583
k^*	516	371	292	295	542	486

TABLE 3 – Estimateurs du maximum de vraisemblance.

3.1.2 Méthode analytique

Pareil qu'avant, tous ces k^* ont passé le test de K-S , avec des p-values toutes plus petites que 10^{-3}, et la majorité sont même plus petites que 10^{-10}.

	AC.PA	ACA.PA	AI.PA	AIR.PA	ALO.PA	BN.PA	BNP.PA	CA.PA
a_1	0.6961	0.6971	0.6960	0.6971	0.6968	0.6957	0.6969	0.9662
b_1	121.6322	105.7932	179.9298	110.2015	133.2713	202.5276	107.6727	131.236
a_2	0.7004	0.6999	0.6969	0.7021	0.6983	0.6965	0.7007	0.6982
b_2	54.6296	61.3794	107.9864	44.6522	71.9825	109.9871	58.1710	82.0256
k^*	302	300	292	298	284	203	302	271

	CAP.PA	COFA.PA	CS.PA	DG.PA	DGM.PA	EDF.PA	EN.PA	GLE.PA
a_1	0.6992	0.6974	0.6957	0.6962	0.7095	0.6962	0.6996	0.6972
b_1	69.1525	101.2627	166.4510	152.0002	26.8455	109.4395	63.8094	92.4274
a_2	0.6975	0.7006	0.6994	0.6998	0.6992	0.7000	0.6959	0.7024
b_2	110.2376	55.1593	69.0412	60.4635	61.0769	56.8085	141.1436	45.9061
k^*	402	302	301	298	328	266	476	301

	HCMLF.PA	HO.PA	IDL.PA	IPN.PA	KER.PA	LR.PA	MC.PA	ML.PA
a_1	0.7018	0.6984	0.7185	0.7010	0.6990	0.6987	0.6984	0.6989
b_1	47.2686	72.0500	34.0685	46.2669	71.9622	77.7054	86.2429	72.0364
a_2	0.6977	0.6961	0.6986	0.6981	0.6976	0.6964	0.6972	0.6971
b_2	91.0608	137.5051	81.2753	84.1360	96.3474	138.3442	113.4174	114.0520
k^*	357	480	5	341	372	364	477	364

	OR.PA	ORA.PA	PUB.PA	RI.PA	RNO.PA	SAF.PA	SAN.PA	SEV.PA
a_1	0.6961	0.6951	0.6995	0.6968	0.6968	0.6966	0.6969	0.6988
b_1	158.7503	189.9411	61.5480	112.0590	80.0460	114.6180	112.6061	77.6070
a_2	0.6973	0.6968	0.6971	0.6960	0.7024	0.7020	0.6956	0.6936
b_2	106.0398	104.1424	126.2913	160.6556	45.2166	45.0211	164.6526	1027.9265
k^*	211	285	491	489	302	305	477	587

	SGO.PA	SU.PA	TTE.PA	VIE.PA	VIV.PA	VK.PA
a_1	0.6970	0.6970	0.6955	0.6962	0.6977	0.7181
b_1	110.8966	131.8521	147.1544	170.9746	93.6896	14.6835
a_2	0.6999	0.6984	0.6998	0.6985	0.7368	0.7029
b_2	65.6845	83.0505	62.3031	76.8390	8.1467	38.2795
k^*	300	290	301	295	687	340

TABLE 4 – Estimateurs par la méthode analytique.

3.2 Graphiques

3.2.1 AC.PA

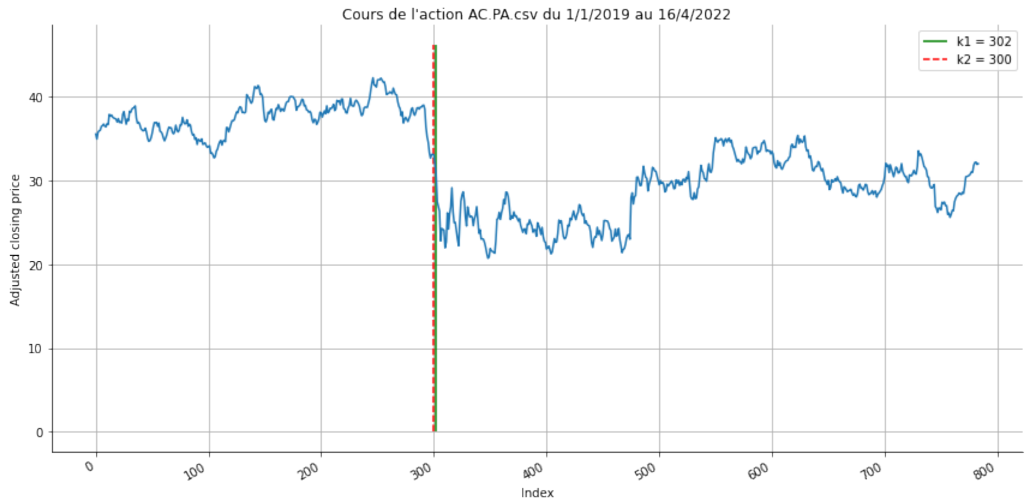

FIGURE 3 – Cours de l'action du AC.PA du 01/01/2019 au 19/01/2022

Les dates des points de rupture trouvées sont le '2020-03-04' et le '2020-03-06'. Ces dates correspondent aux conséquences de la pandémie mondiale due au Covid-19 à savoir l'annonce du premier confinement, qui marque une forte baisse d'activité.

3.2.2 ACA.PA

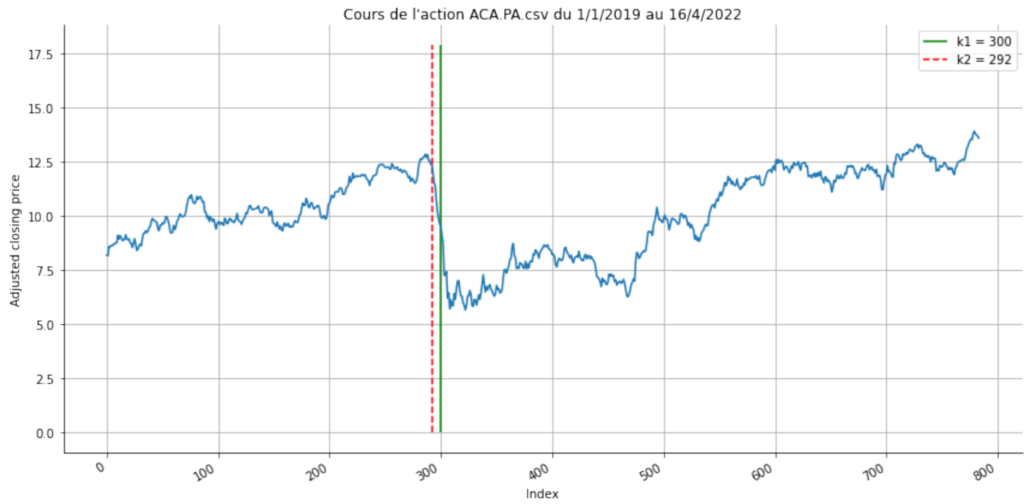

FIGURE 4 – Cours de l'action du ACA.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-03-04'. Ces dates correspondent à la monté de la pandémie du Covid-19 et aux conséquences sur l'économie mondiale

3.2.3 AI.PA

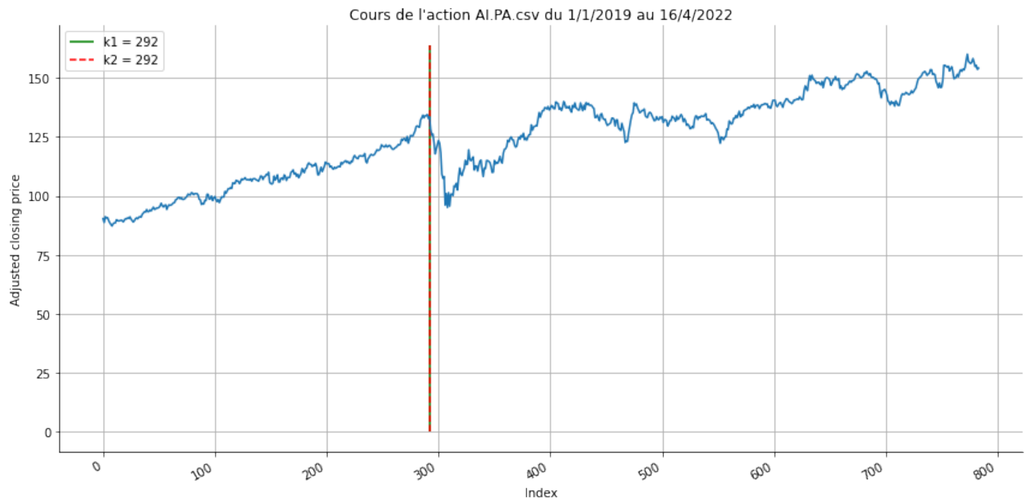

FIGURE 5 – Cours de l'action du AI.PA du 01/01/2019 au 19/01/2022

Le point de rupture trouvé a eu lieu le '2020-02-21'. Cette date renvoie au Covid-19 et à ses conséquences sur l'économie mondiale.

3.2.4 AIR.PA

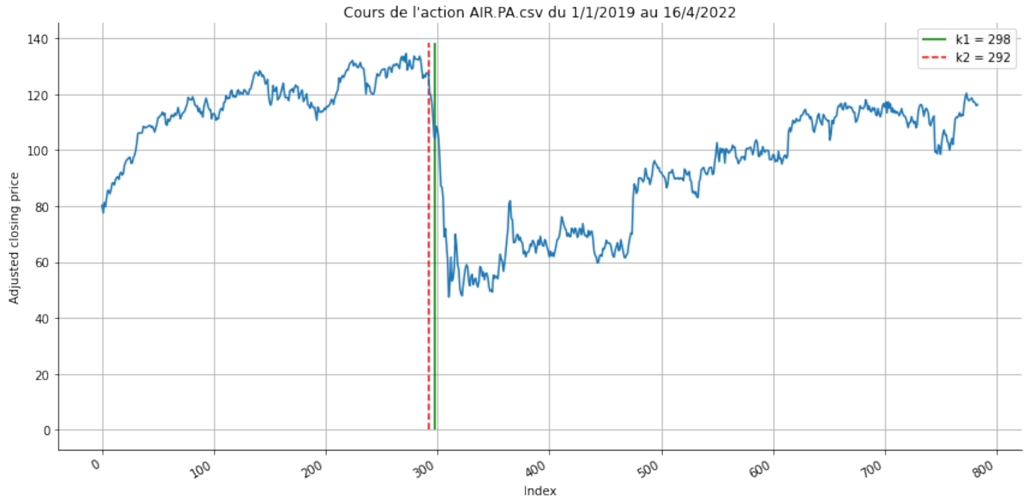

FIGURE 6 – Cours de l'action du AIR.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-03-02' et le '2020-02-21'. Ces dates renvoie au Covid-19 et à ses conséquences sur l'économie mondiale.

3.2.5 ALO.PA

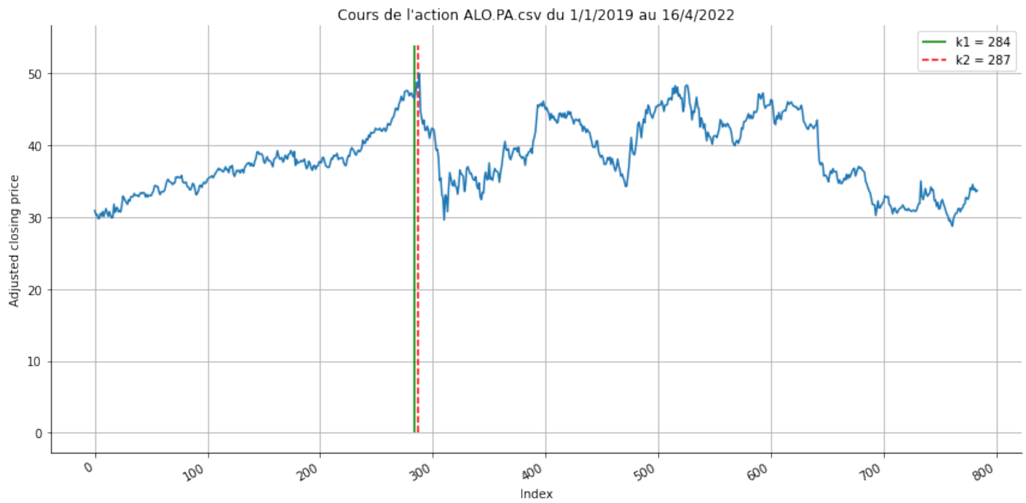

FIGURE 7 – Cours de l'action du ALO.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-03-02'. Ces dates renvoie au Covid-19 et à ses conséquences sur l'économie mondiale.

3.2.6 BN.PA

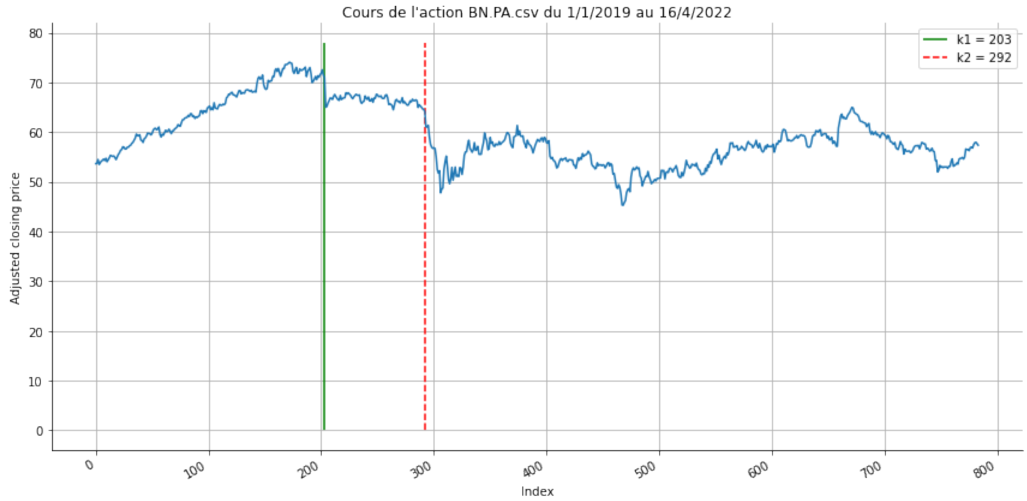

FIGURE 8 – Cours de l'action du BN.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2019-10-17' et le '2020-02-21'. Le '2019-10-17', le cours de Danone a chuté en raison de l'annonce de ses résultats trimestriels qui étaient inférieurs aux attentes. D'autre part Danone a revu à la baisse les prévisions de son chiffre d'affaire pour le quatrième trimestre de 2019. Le '2020-02-21' correspond à l'épidémie du Covid-19 et de ses conséquences sur l'économie mondiale.

3.2.7 BNP.PA

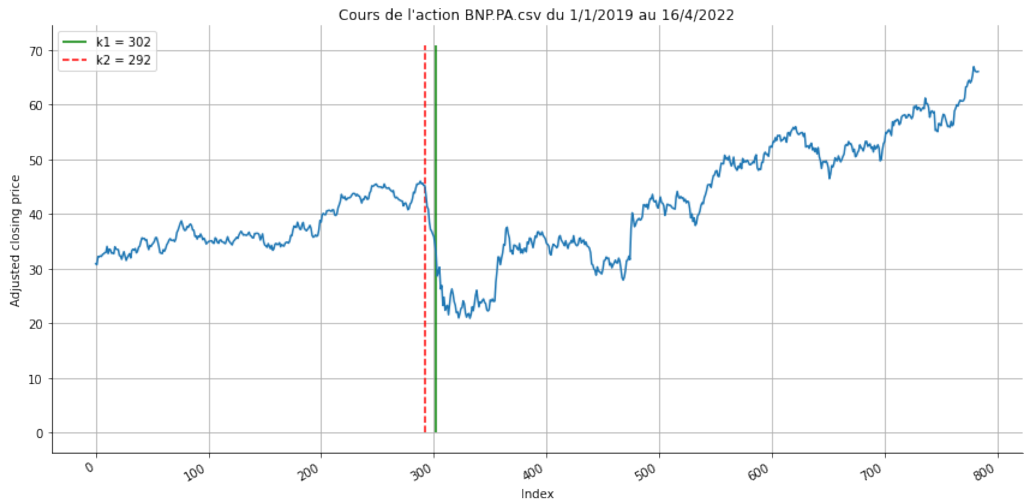

FIGURE 9 – Cours de l'action du BNP.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-03-06'. Ces dates correspondent au Covid-19 et à ses conséquences sur l'économie mondiale.

3.2.8 CA.PA

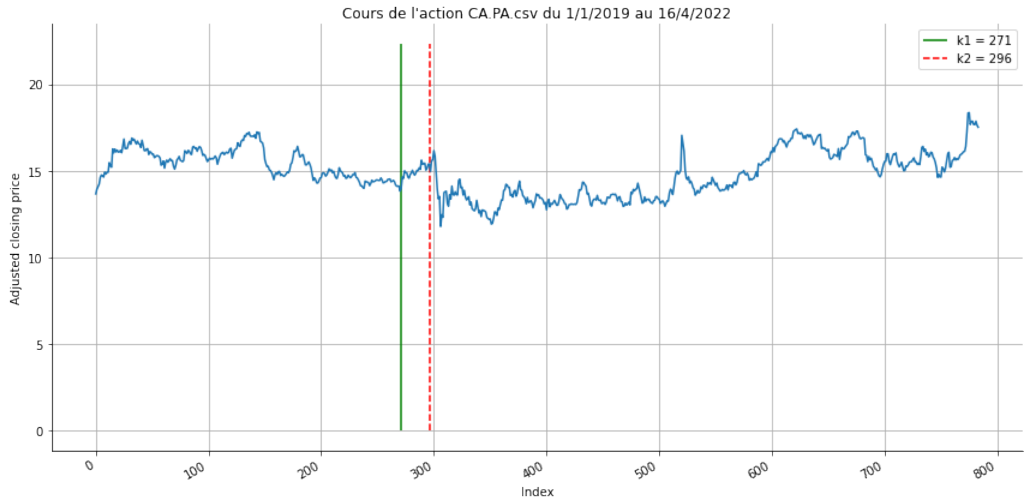

Figure 10 – Cours de l'action du CA.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-01-23' et le '2020-02-27'. Le '2020-02-27' correspond au Covid-19 et à ses conséquences sur l'économie mondiale.

3.2.9 CAP.PA

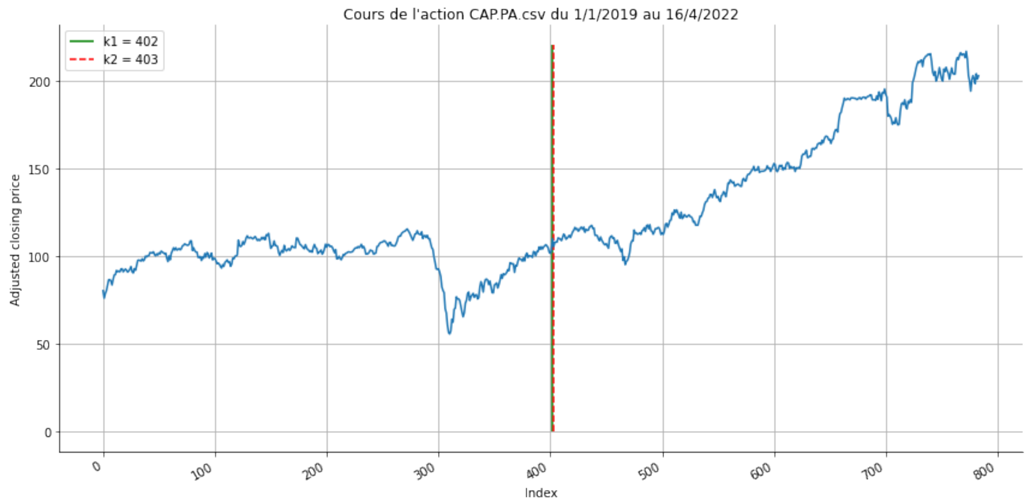

Figure 11 – Cours de l'action du CAP.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent quasiment le même temps de rupture..

Les points de rupture trouvés ont eu lieu le '2020-07-29' et le '2020-07-30'. Ces dates correspondent à la levée de certaines restrictions en France.

3.2.10 COFA.PA

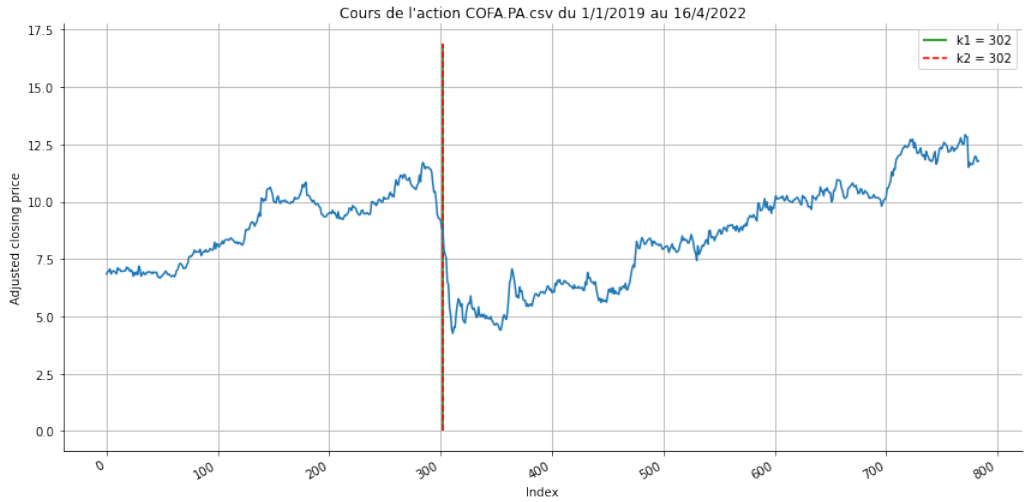

Figure 12 – Cours de l'action du COFA.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 302$).

Le point de rupture trouvé a eu lieu le '2020-03-06'. Cette date correspond au début des restrictions des libertés dues au Covid-19.

3.2.11 CS.PA

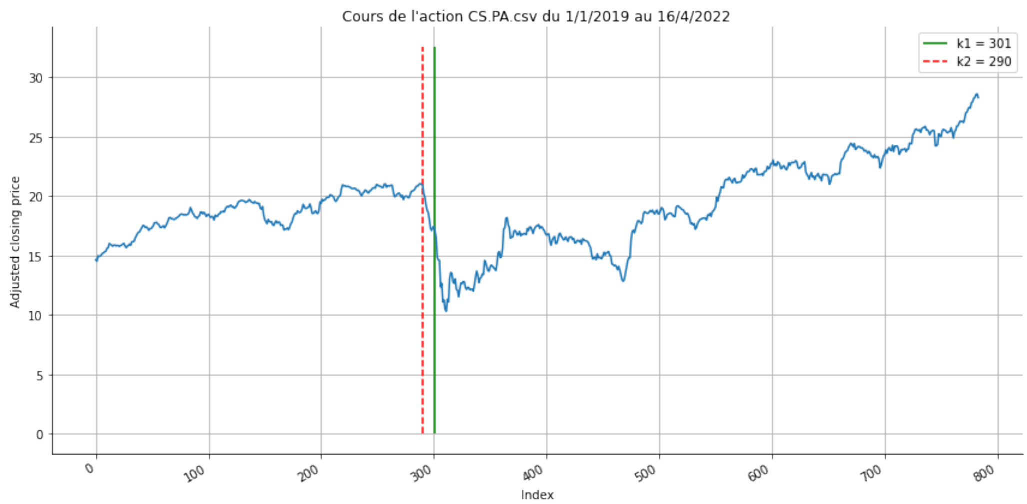

Figure 13 – Cours de l'action du CS.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-02-19' et le '2020-03-05'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.12 DG.PA

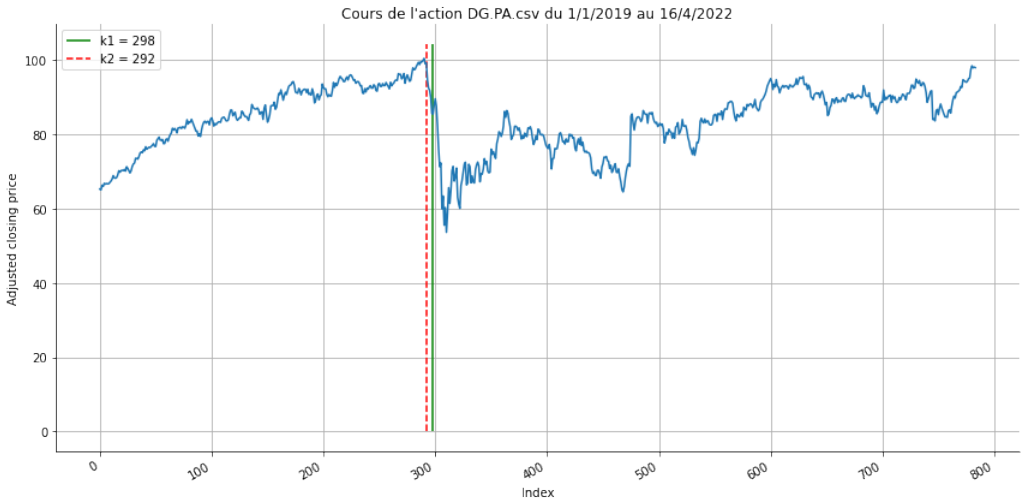

FIGURE 14 – Cours de l'action du DG.PA du 01/01/2019 au 19/01/2022

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-03-02'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.13 DGM.PA

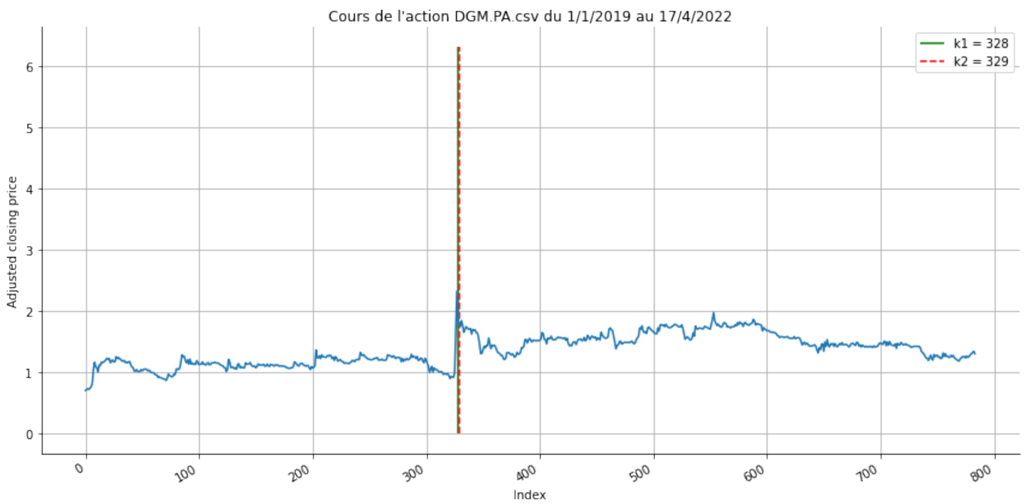

FIGURE 15 – Cours de l'action du DGM.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent quasiment le même temps de rupture.

Les points de rupture trouvés ont eu lieu le '2020-04-15' et le '2020-04-16'.

3.2.14 EDF.PA

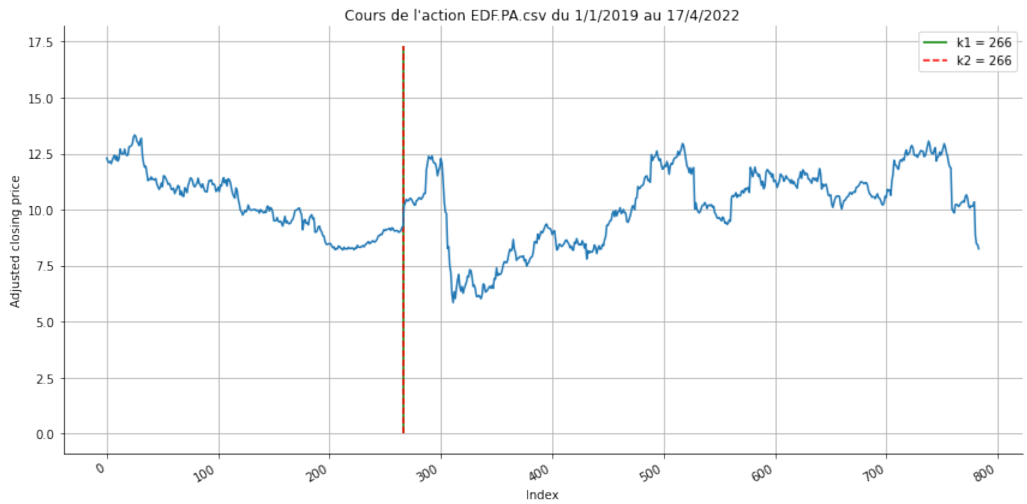

FIGURE 16 – Cours de l'action du EDF.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 266$).

Le point de rupture trouvé a eu lieu le '2020-01-16'. Cette date correspond à l'annonce des résultats du quatrième trimestre de 2019. EDF a eu un bénéfice net multiplié par plus de quatre en 2019, ce qui explique l'augmentation de son cours boursier.

3.2.15 EN.PA

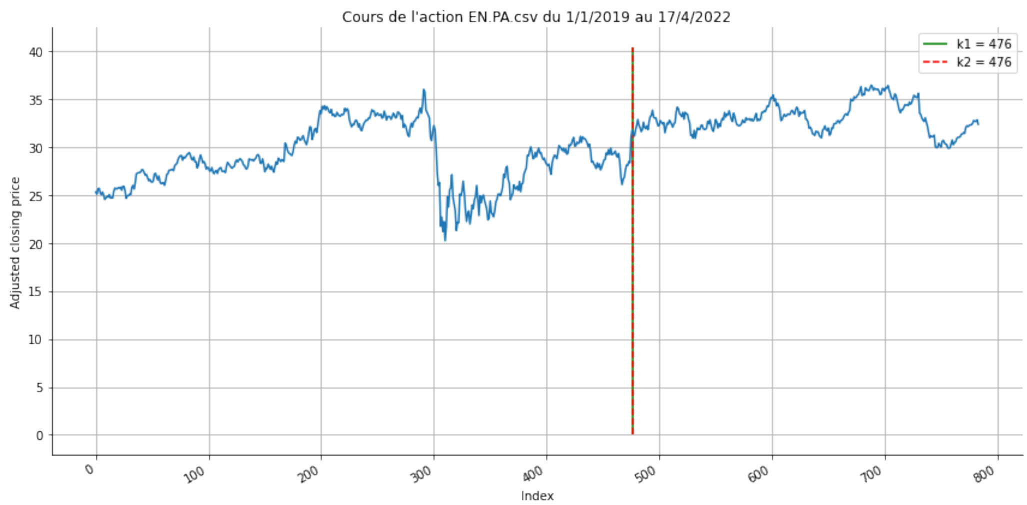

FIGURE 17 – Cours de l'action du EN.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 476$).

Le point de rupture trouvé a eu lieu le '2020-11-10'. Bouygues a vu son chiffre d'affaire augmenté de 10%. Après les annonces de ses résultats son action a augmenté.

3.2.16 GLE.PA

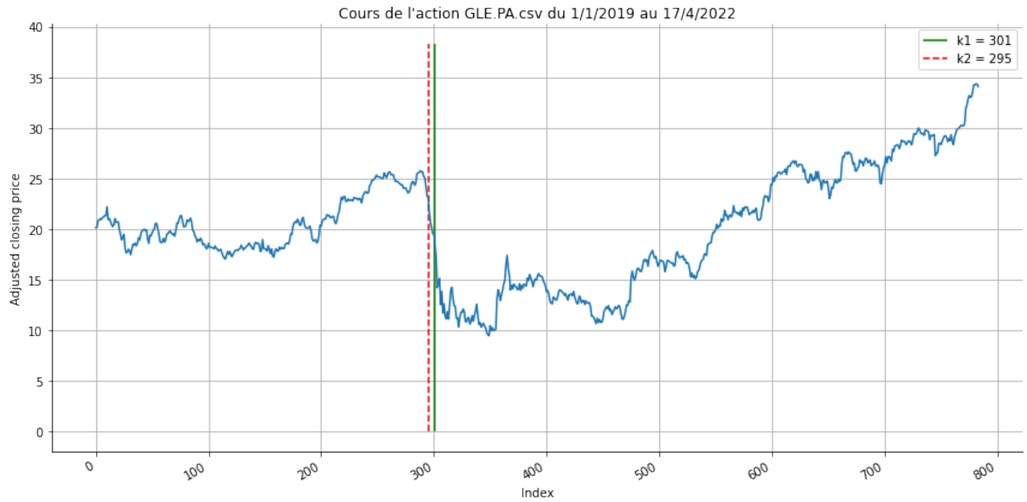

FIGURE 18 – Cours de l'action du GLE.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent quasiment le même temps de rupture (différence de 6).

Les points de rupture trouvés ont eu lieu le '2020-03-05' et le '2020-02-26'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.17 HCMLF.PA

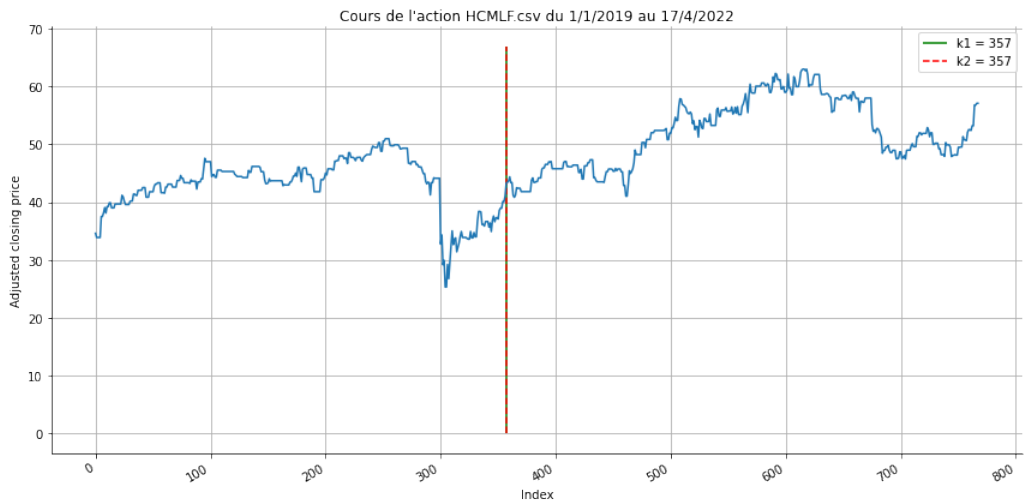

FIGURE 19 – Cours de l'action HCMLF.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 357$).

Le point de rupture trouvé a eu lieu le '2020-05-27'.

3.2.18 HO.PA

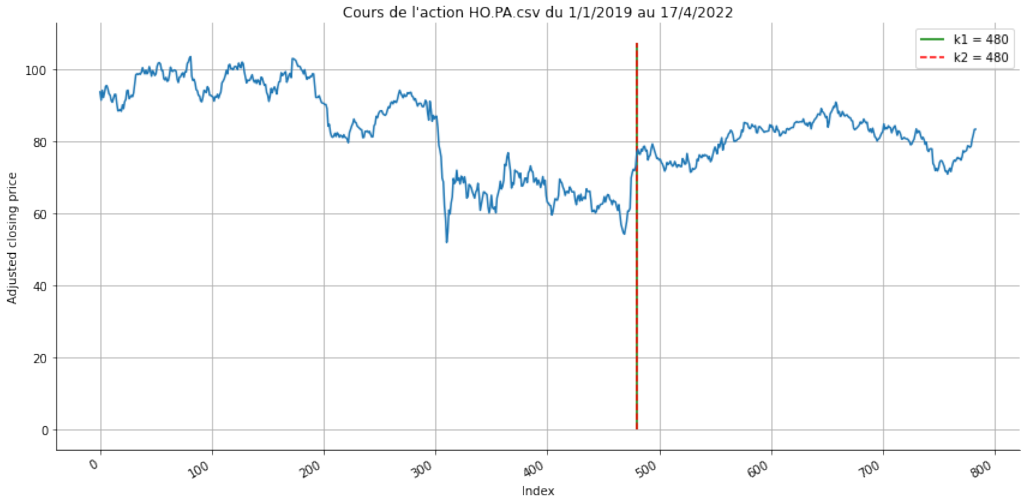

FIGURE 20 – Cours de l'action du HO.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 480$).

Le point de rupture trouvé a eu lieu le '2020-11-16'.

3.2.19 IDL.PA

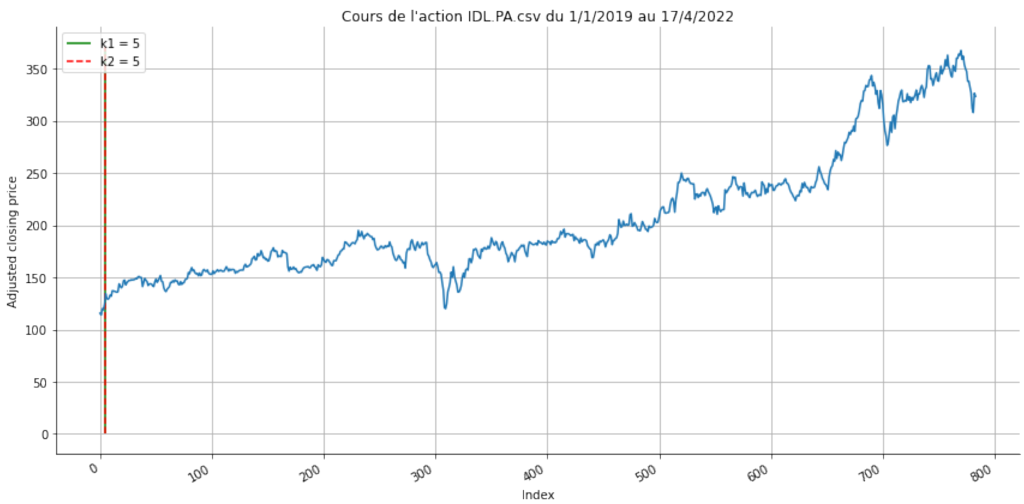

FIGURE 21 – Cours de l'action du IDL.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent le même temps de rupture (la même valeur de $k = 5$).

Le point de rupture trouvé a eu lieu le '2019-01-09'.

3.2.20 IPN.PA

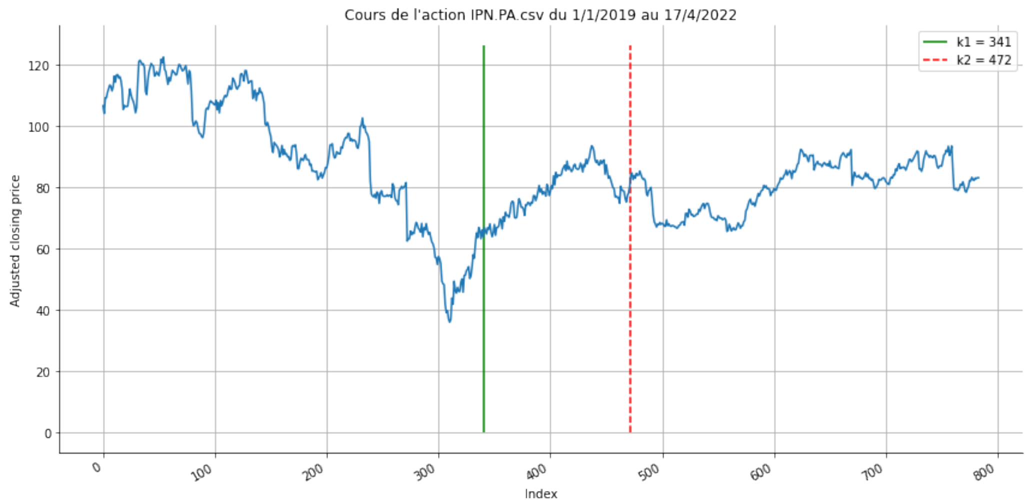

FIGURE 22 – Cours de l'action du IPN.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent des temps de rupture différents (Une différence de 131 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2020-11-04' et le '2020-05-05'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.21 KER.PA

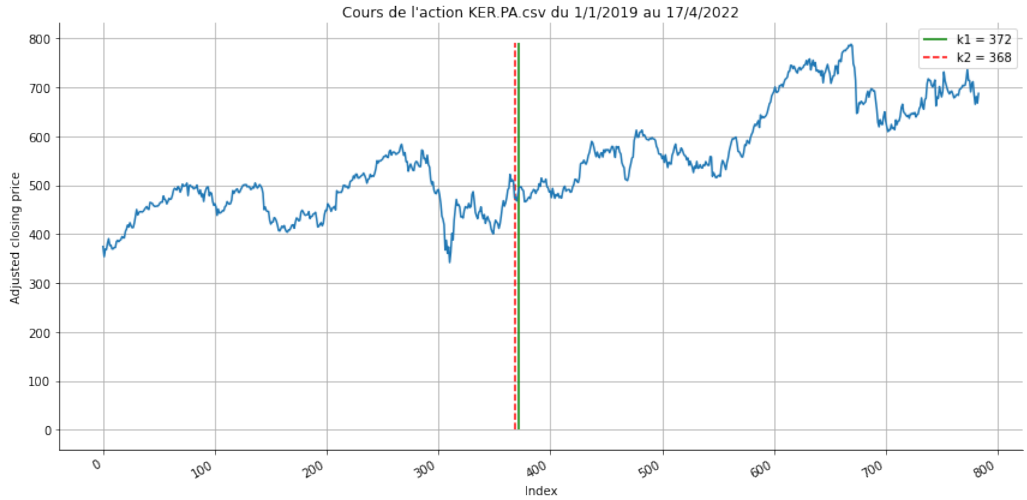

Figure 23 – Cours de l'action du KER.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent des temps de rupture très proches (une différence de 4 entre les deux valeurs de k).

Cette rupture est probablement due à la pandémie mondiale du Covid 19.

Les points de rupture trouvés ont eu lieu le '2020-06-11' et le '2020-06-17'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.22 LR.PA

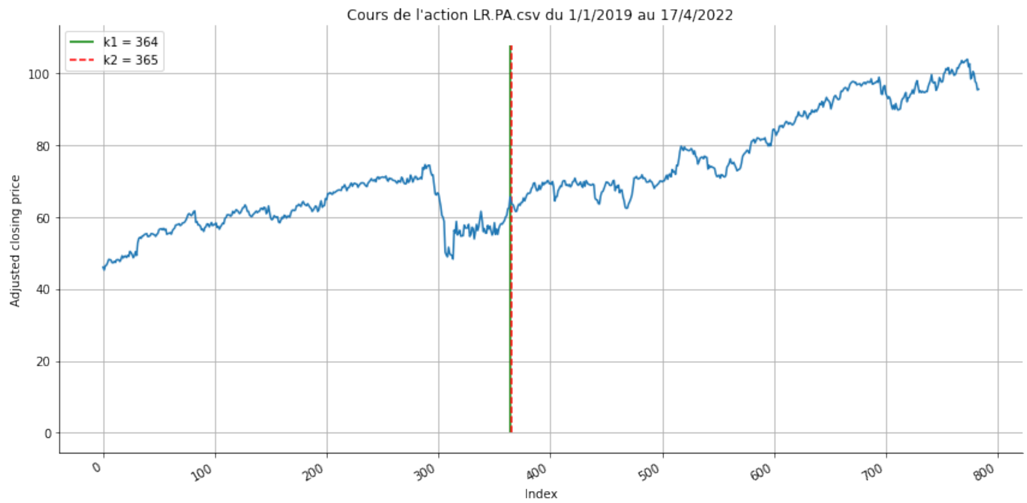

Figure 24 – Cours de l'action du LR.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 1 entre les deux valeurs de k).

Les points de rupture trouvés ont eu lieu le '2020-06-08' et le '2020-06-05'.

3.2.23 MC.PA

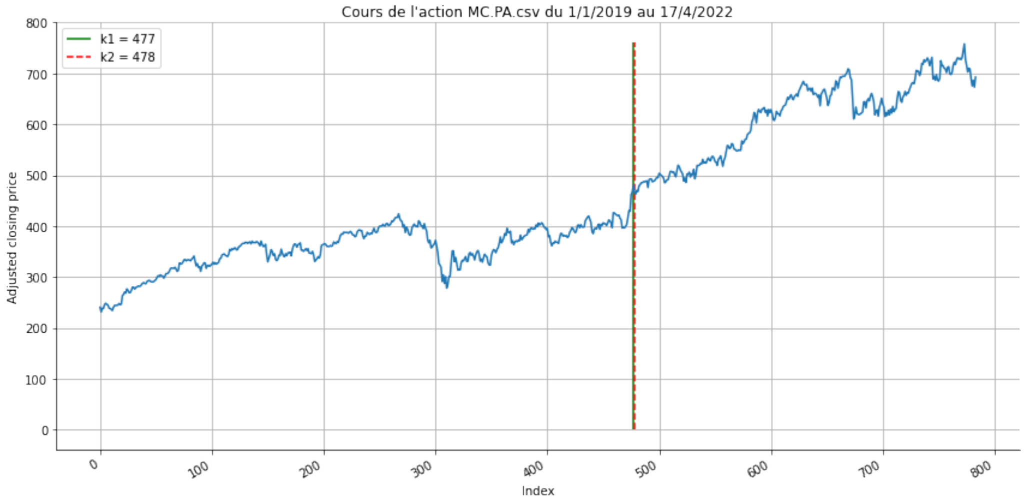

FIGURE 25 – Cours de l'action du MC.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise aussi, les deux méthodes analytique, et log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 1 entre les deux valeurs de k).

Les points de rupture trouvés ont eu lieu le '2020-11-12' et le '2020-11-17'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.24 ML.PA

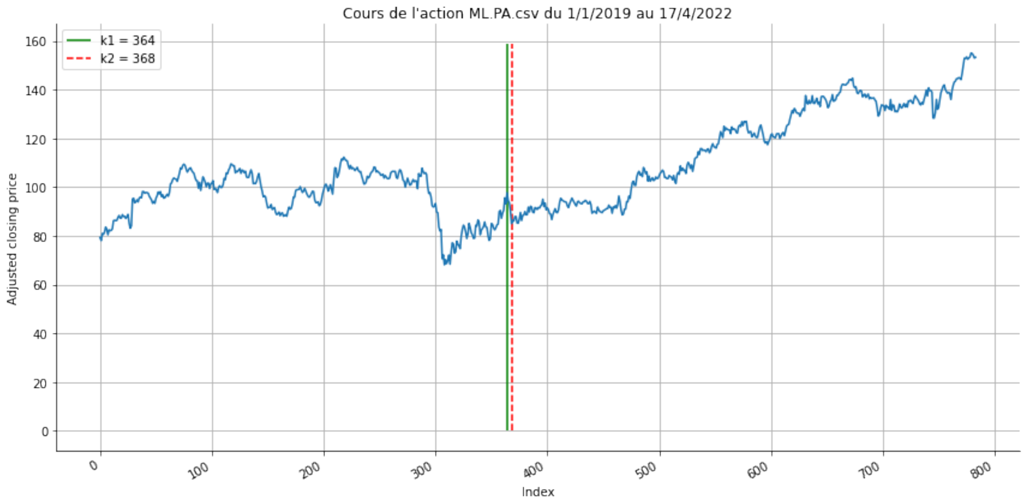

FIGURE 26 – Cours de l'action du ML.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytiques, et log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 4 entre les deux valeurs de k , la première est à 364 et la deuxième est à 368).

Les points de rupture trouvés ont eu lieu le '2020-06-11' et le '2020-06-05'.

3.2.25 OR.PA

FIGURE 27 – Cours de l'action du OR.PA du 01/01/2019 au 19/01/2022

On peut voir que pour cette entreprise, on a la même valeur de k, que ça soit pour la méthode analytique ou bien la log-vraisemblance, et cette valeur est égale à 211.

Le point de rupture trouvé a eu lieu le '2019-10-29'.

3.2.26 ORA.PA

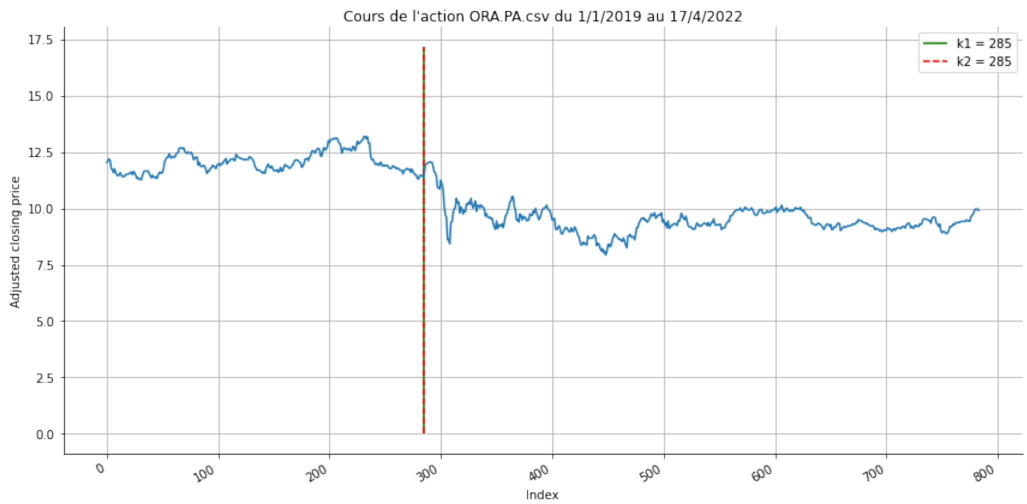

FIGURE 28 – Cours de l'action du ORA.PA du 01/01/2019 au 19/01/2022

On peut voir que pour cette entreprise, on a la même valeur de k, que ça soit pour la méthode analytique ou bien la log-vraisemblance, et cette valeur est égale à 285.

Le point de rupture trouvé a eu lieu le '2020-02-12'.

3.2.27 PUB.PA

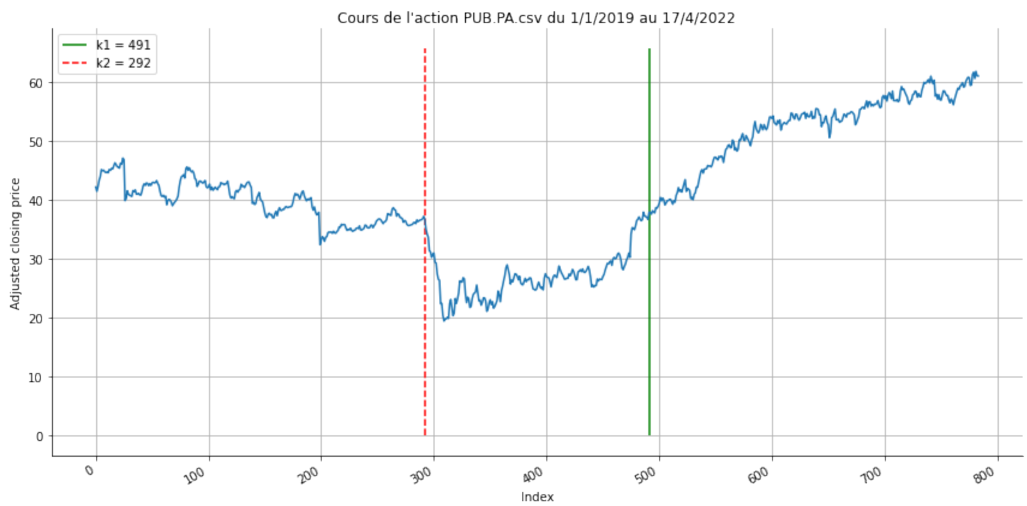

FIGURE 29 – Cours de l'action du PUB.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent des temps de rupture différents (Une différence de 199 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-12-01'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.28 RI.PA

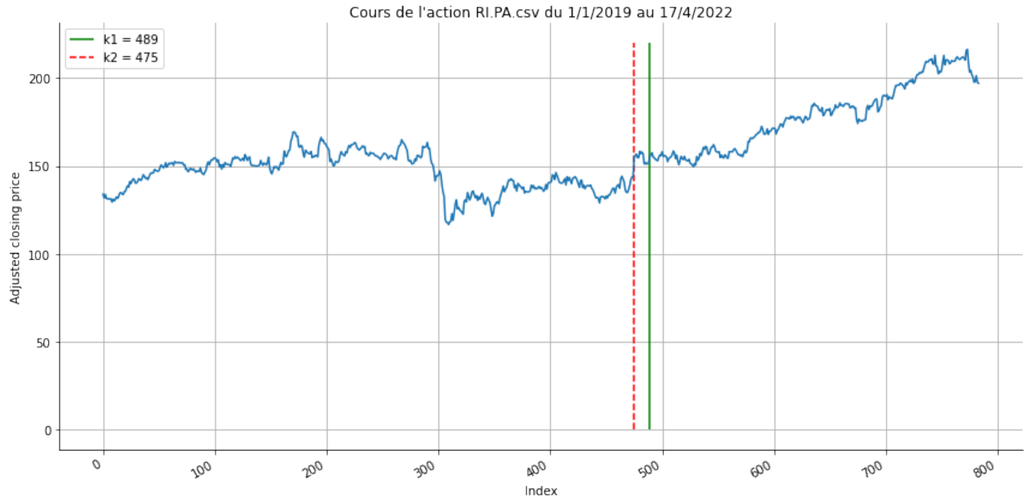

FIGURE 30 – Cours de l'action du RI.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, la différence de k entre la méthode analytique et la méthode de log-vraisemblance n'est pas trop grande, et cette valeur est donc 14.

Les points de rupture trouvés ont eu lieu le '2020-11-09' et le '2020-11-27'. Ces dates correspondent aux conséquences de la pandémie du Covid-19 sur l'économie mondiale.

3.2.29 RNO.PA

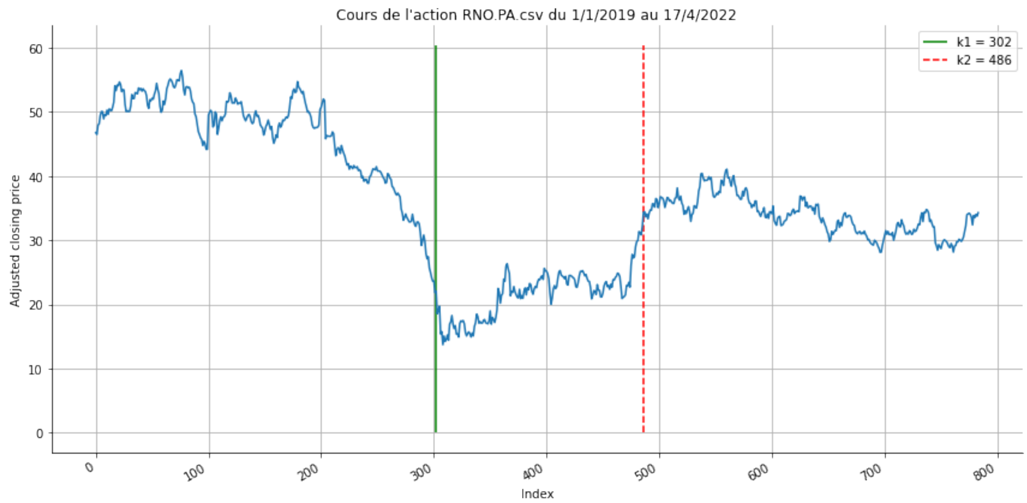

FIGURE 31 – Cours de l'action du PNO.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique, et log-vraisemblance nous donnent des temps de rupture différents (Une différence de 184 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2020-11-24' et le '2020-03-06'.

3.2.30 SAF.PA

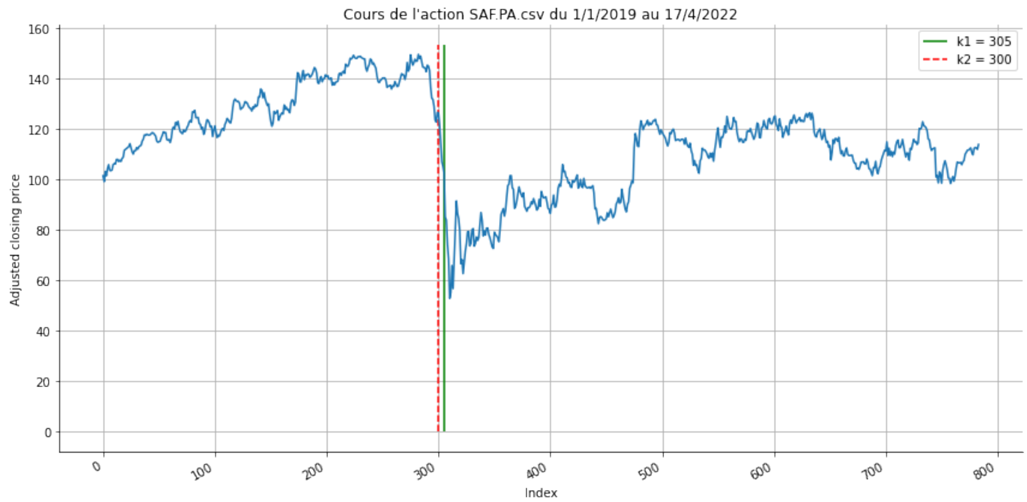

FIGURE 32 – Cours de l'action du SAF.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytiques, et log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 5 entre les deux valeurs de k , la première est a 305 et la deuxième est à 300).

Les points de rupture trouvés ont eu lieu le '2020-03-04' et le '2020-03-11'.

3.2.31 SAN.PA

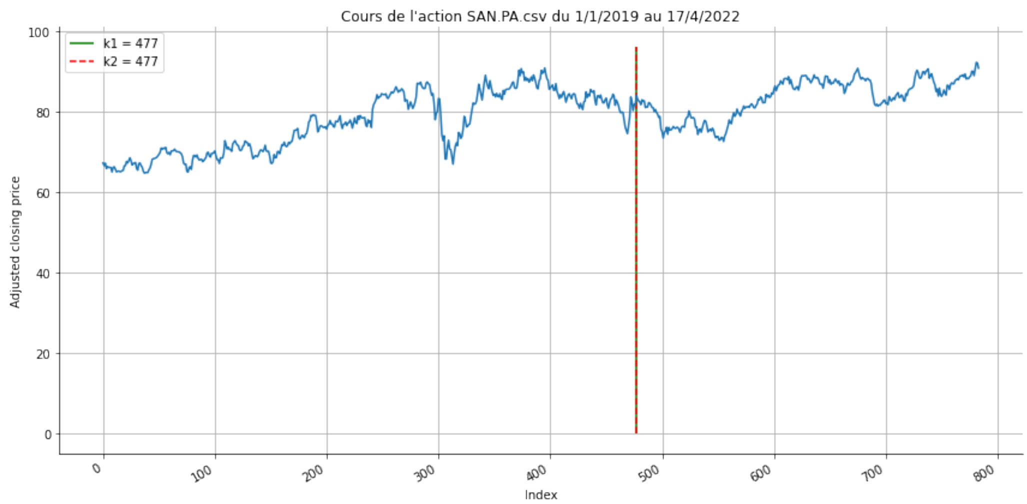

FIGURE 33 – Cours de l'action du SAN.PA du 01/01/2019 au 19/01/2022

Le point de rupture trouvé a eu lieu le '2020-11-11'.

3.2.32 SEV.PA

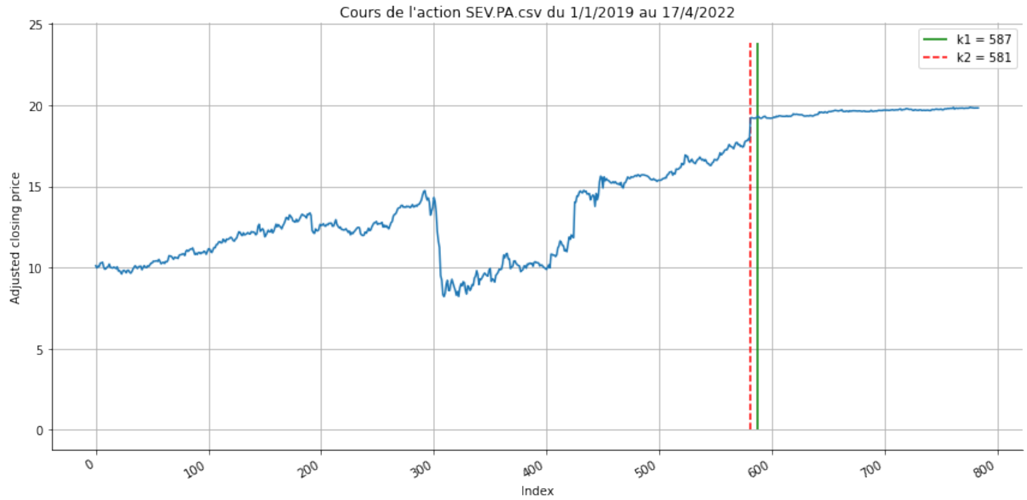

FIGURE 34 – Cours de l'action du SEV.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 6 entre les deux valeurs de k).

Les points de rupture trouvés ont eu lieu le '2021-04-12' et le '2021-04-20'.

3.2.33 SGO.PA

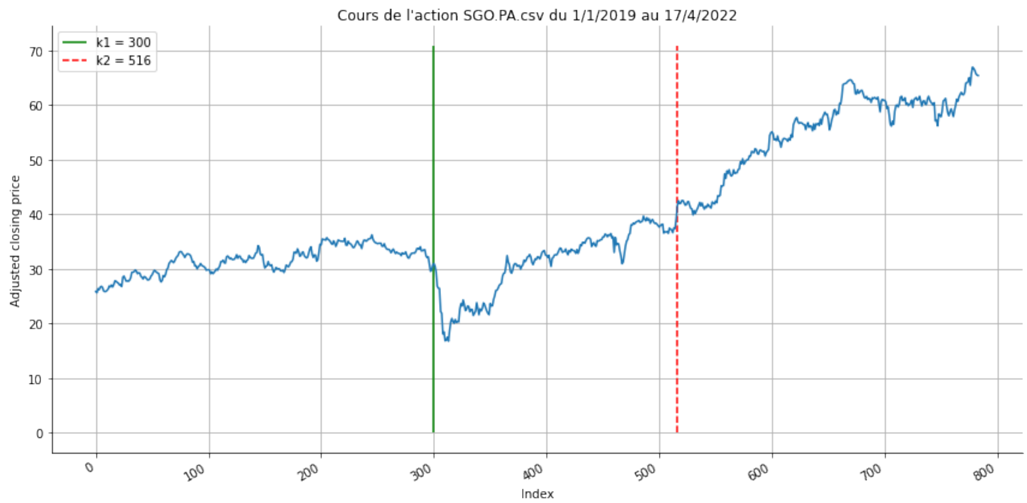

FIGURE 35 – Cours de l'action du SGO.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent des temps de rupture différents (une différence de 216 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2021-01-07' et le '2020-03-04'.

3.2.34 SU.PA

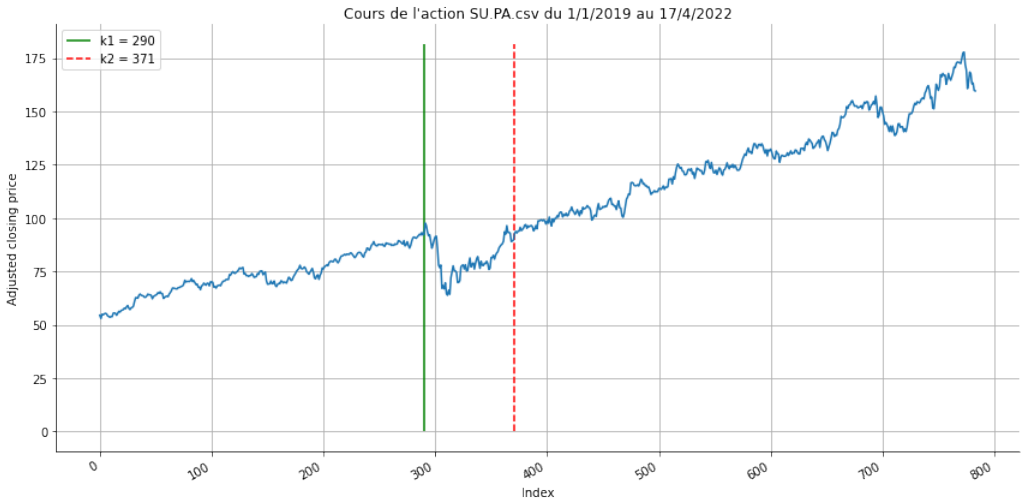

Figure 36 – Cours de l'action du SU.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent des temps de rupture différents (une différence de 81 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2020-06-16' et le '2020-02-19'.

3.2.35 TTE.PA

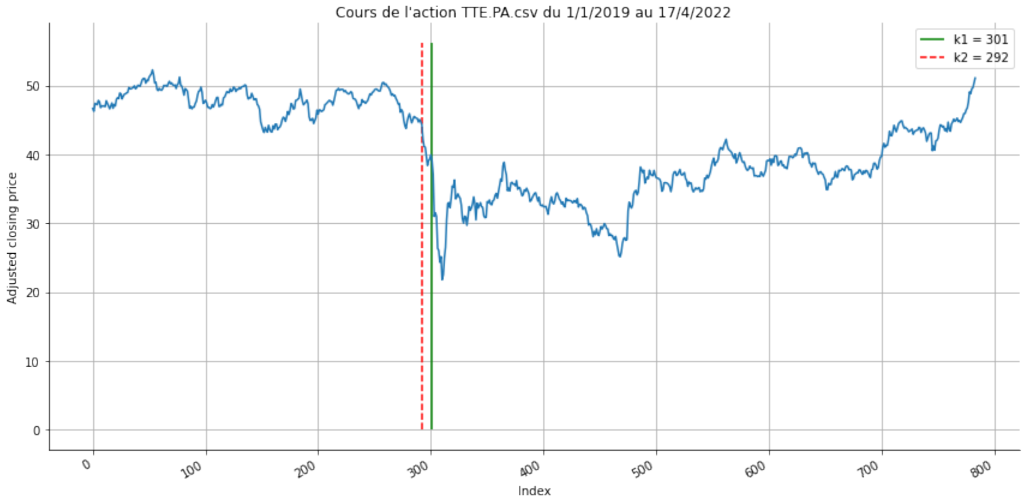

Figure 37 – Cours de l'action du TTE.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent quasiment le même temps de rupture (une différence de 9 entre les deux valeurs de k).

Les points de rupture trouvés ont eu lieu le '2020-02-21' et le '2020-03-05'.

3.2.36 VIE.PA

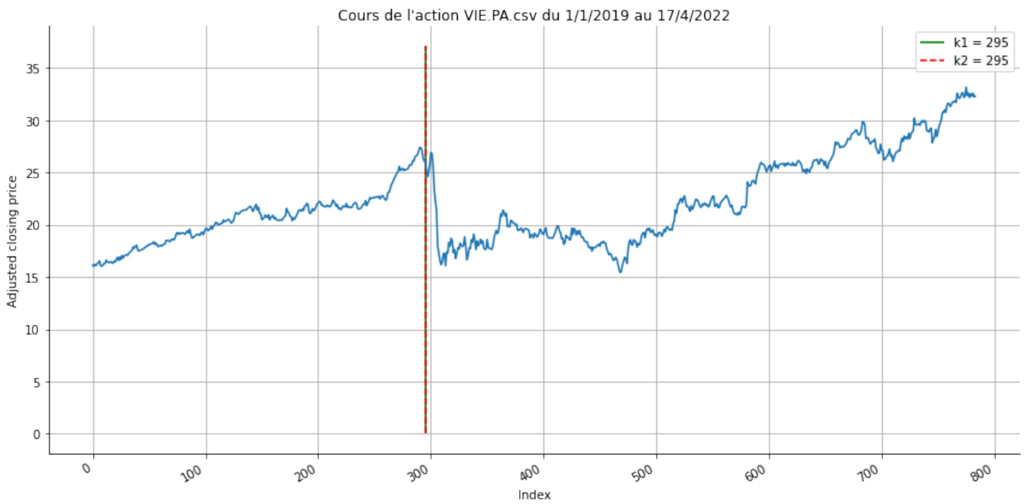

Figure 38 – Cours de l'action du VIE.PA du 01/01/2019 au 19/01/2022

Pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent le même temps de rupture (les valeurs de k sont égales).

Le point de rupture trouvé a eu lieu le '2020-02-26'.

3.2.37 VIV.PA

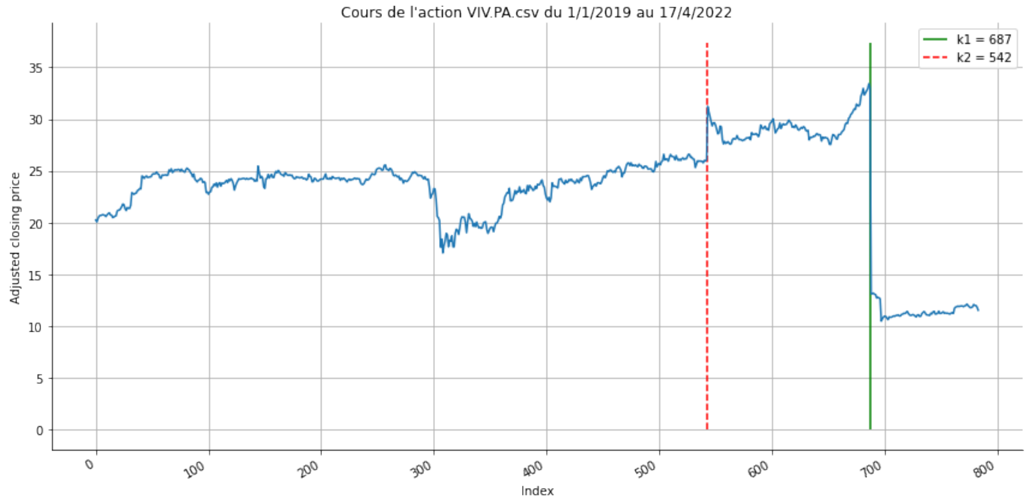

FIGURE 39 – Cours de l'action du VIV.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent des temps de rupture différents (une différence de 145 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2021-02-12' et le '2021-09-07'.

3.2.38 VK.PA

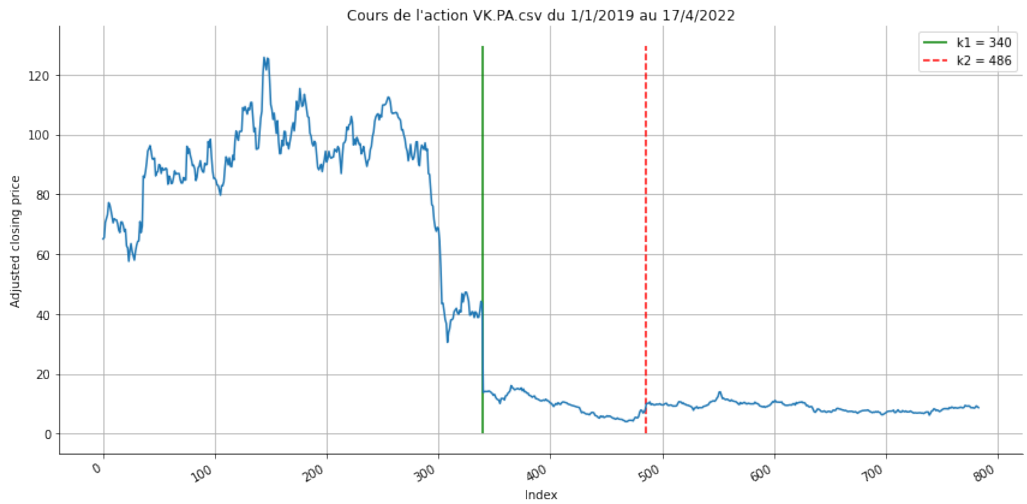

FIGURE 40 – Cours de l'action du VK.PA du 01/01/2019 au 19/01/2022

Nous remarquons que pour cette entreprise, les deux méthodes analytique et de log-vraisemblance nous donnent des temps de rupture différents (une différence de 146 pour la valeur de k).

Les points de rupture trouvés ont eu lieu le '2020-11-24' et le '2020-05-04'.

4 Conclusion et Perspectives

La méthode mise en place permet de trouver le point de rupture le plus significatif sur un échantillon donné. S'il en existe plusieurs et que l'on souhaite tous les trouver, on pourrait appliquer à nouveau cette méthode de la manière suivante :

— On trouve le plus grand point de rupture sur l'échantillon de départ E, qui se divise donc en deux échantillons E_1 et sur E_2. On effectue à nouveau la méthode sur E_1 et E_2.

— On applique la méthode pour trouver le plus grand point de rupture sur chaque sous-échantillon. Tant que l'on en trouve un, chaque sous-échantillon est divisé en deux.

— Si l'on ne trouve pas de nouveau point de rupture sur un échantillon E_i donné, alors le test d'adéquation de Kolmogorov-Smirnov est mis en oeuvre pour s'assurer que E_i suit bien une loi de Weibull dont les paramètres sont estimés.

Dans ce projet, on a réalisé deux méthodes pour trouver le point de rupture le plus significatif sur nos données : la méthode du maximum de vraisemblance et la méthode analytique qui transforme le modèle de la loi de Weibull en modèle linéaire simple. Tous les ruptures (k^*) trouvées ont passé le test de Kolmogorov-Smirnov d'homogénéité, avec l'hypothèse "H_0 : $X_1 ... X_k$ et $X_{k+1} ... X_n$ suivent la même loi", et toutes les p-values plus petites que 10^{-3} . Cela implique que toutes les ruptures (k^*) trouvées sont significatives.

Références

[1] Dariush Ghorbanzadeh, Philippe Durand, Luan Jaupi. An Analytical Method for Detecting the Change-Point in Simple Linear Regression Model. Application at Weibull Distribution. *Journal of Applied Quantitative Methods*, Association for Development through Science and Education, Romania, 2016, 11 (1), pp. 1-13. ⟨hal-02464929⟩ https://hal-cnam.archives-ouvertes.fr/hal-02464929/document

[2] William H. Press and Saul A. Teukolsky. Kolmogorov-Smirnov Test for Two-Dimensional Data. https://aip.scitation.org/doi/pdf/10.1063/1.4822753

[3] Jean-Jacques Ruch. Statistique : Tests d'hypothèses. https://www.math.u-bordeaux.fr/~mchabano/Agreg/ProbaAgreg1213-COURS3-Stat2.pdf

[4] Quandt, R. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. Journal of the American Statistical Association, 53, 873-880.

[5] Bernard, A., Bosi-Levenbach, E.C. (1953). The plotting of observations on probability paper. Stat. Neederlandica, 7, 163-173.

A Annexe : Codes

A.1 Modèle de détection de ruptures

```python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import minimize
from functools import partial
import warnings
warnings.filterwarnings("ignore")
from tqdm import tqdm_notebook as tqdm
import math
import datetime
import os

"""> importation des données du CAC40"""

Liste_donnees=os.listdir(r'CAC40\DonCAC40\\')

data = []
for i in range(len(Liste_donnees)):
    F='CAC40/DonCAC40\\'+Liste_donnees[i]
    a = pd.read_csv(F, delimiter=',')
    a=a[(a['Date'] >='2019-01-01')]
    a = a.reset_index()
    a=a.dropna()  # supprimer les données manquantes
    data.append(a[a.columns[7]])

data_time = []

for i in range(len(Liste_donnees)):
    F='CAC40/DonCAC40\\'+Liste_donnees[i]
    a = pd.read_csv(F, delimiter=',')
    a=a[(a['Date'] >='2019-01-01')]
    a = a.reset_index()

    a=a.dropna()  # supprimer les données manquantes
    data_time.append(a[a.columns[1]])

#La fonction Y de notre variable cible
def Y(data):
    Z=[np.log(1+data[i]/data[i-1]) for i in range(1,len(data))]
    return(Z)

"""## Code méthode analytique"""
```

```
def estimationAB(X):

    n=len(X)
    hbar=np.zeros(n)
    Ybar=np.zeros(n)
    hbar_etoile=np.zeros(n)
    Ybar_etoile=np.zeros(n)
    A1_chap_nom=np.zeros(n)
    A2_chap_nom=np.zeros(n)
    A1_chap_denom=np.zeros(n)
    A2_chap_denom=np.zeros(n)
    A1_chap=np.zeros(n)
    A2_chap=np.zeros(n)
    B1_chap=np.zeros(n)
    B2_chap=np.zeros(n)


    for k0 in range(4,n-4):
        Y=F(k0,n)
        hbar[k0]=(1/k0)*sum(np.log(np.array(sorted(X[:k0]))))
        Ybar[k0]=(1/k0)*sum(Y[:k0])
        hbar_etoile[k0]=(1/(n-k0))*sum(np.log(np.array(sorted(X[k0:]))))
        Ybar_etoile[k0]=(1/(n-k0))*sum(np.array(Y[k0:]))

        #estimation des A1 B1 A2 B2

        #estimation A1
        A1_chap_nom[k0]=sum((np.log(np.array(sorted(X[:k0])))-
            ↪ hbar[k0])*(np.array(Y[:k0])-np.array(Ybar[k0])))
        ↪ A1_chap_denom[k0]=sum((np.log(np.array(sorted(X[:k0])))-hbar[k0])**2)
        A1_chap[k0]=A1_chap_nom[k0]/A1_chap_denom[k0]

        #estimation B1
        B1_chap[k0]=Ybar[k0]-A1_chap[k0]*hbar[k0]

        #estimation A2
        A2_chap_nom[k0]=sum((np.log(np.array(sorted(X[k0:])))-
            ↪ hbar_etoile[k0])*(np.array(Y[k0:])-np.array(Ybar_etoile[k0])))
        A2_chap_denom[k0]=sum((np.log(sorted(X[k0:]))-hbar_etoile[k0])**2)
        A2_chap[k0]=A2_chap_nom[k0]/A2_chap_denom[k0]

        #estimation B2
        B2_chap[k0]=Ybar_etoile[k0]-A2_chap[k0]*hbar_etoile[k0]

    result=[A1_chap,B1_chap,A2_chap,B2_chap]
```

```python

    return result

def ln(x):
    return np.log(x)
def MR(i,k0,n):
    if i <= k0:
        return (i-0.3)/(k0+0.4)
    else :
        return (i-k0-0.3)/(n-k0+0.4)

def F(k0,n):
    Zt=[ ln(-ln(1-MR(i,k0,n))) for i in range(1,n+1)]
    return Zt

#Estimation des A1 B1 A2 B2 finaux

#Fonction qui calcul K*
def Calcul_K_Etoile(B1,A1,B2,A2,X):
     #Fonction qui renvoie le vecteur des Xhi_i,j(k0)

    def Calcul_Vect_E(k0,n,B1_k0,A1_k0,B2_k0,A2_k0,Y,X):
        Vect_E_k0=[]
        for i in range(0,k0):
            X1=sorted(X[:k0])
            Vect_E_k0.append(Y[i]-B1_k0-A1_k0*np.log(X1[i]))
        for i in range(k0,n):
            X2=sorted(X[k0:])
            Vect_E_k0.append(Y[i]-B2_k0-A2_k0*np.log(X2[i-k0]))
        return(Vect_E_k0)

#calcul de D avec k0, n et Vect_E_k0 qui représente un vecteur de taille n, où les k0 premier éléments correspondent aux Xhi_1,i(k0)
#et les k0+1 à n représentent les Xhi_2,i(k0)

    def Calcul_D(k0,n,Vect_E_k0):
        D=0
        for i in range(0,k0)  :    ## i va de 0 à k0-1, donc on récupère les éléments de vect_E de i allant de 1 à k0
            D=D+(Vect_E_k0[i])**2
        for i in range(k0,n):
            D=D+(Vect_E_k0[i])**2

        return(D)

    n=len(X)
    D=np.zeros(n-8)  #Pour avoir un vecteur D de la bonne taille
    for i in range(4,n-4):
        Y=F(i,n)
```

```
        D[i-4]=Calcul_D(i,n,Calcul_Vect_E(i,n,B1[i],A1[i],B2[i],A2[i],Y,X))

    k_etoile=np.argmin(D)+4
    return k_etoile


def estimation_finaleAB(X):

    A1=estimationAB(X)[0]
    B1=estimationAB(X)[1]
    A2=estimationAB(X)[2]
    B2=estimationAB(X)[3]
    k=Calcul_K_Etoile(B1,A1,B2,A2,X)

    result=[A1[k],B1[k],A2[k],B2[k]]
    return result

def First_Method_analytic(X):
    [A1t,B1t,A2t,B2t]=estimationAB(X)
    k=Calcul_K_Etoile(B1t,A1t,B2t,A2t,X)
    [A1,B1,A2,B2]=estimation_finaleAB(X)
    param1={"a1": math.exp(-B1/A1),"b1": A1,"a2": math.exp(-B2/A2),"b2":
    ↪  A2, "k*": k}
    return param1

"""## Code log Vraisemblance"""

#fonction qui estime le moment de rupture ainsi que les 4 parametres
↪  a1,a2,b1,b2 en fonction des données ainsi  estime k

def Second_Method_log_vrais(df):
    n = len(df)
    taille = np.infty
    Vect0 = [1,1,1,1]
    k = 0

    #On retourne log vrai semblance de la fonction
    def funct_2(df,k,param):
        return funct_1(df[:k],param[:2])+funct_1(df[k:],param[2:])


    def funct_1(df,param):
        taille = len(df)
        a1 , b1 = param[1] , param[0]
        coef = taille*np.log(b1) - b1*taille*np.log(a1)
        coef11 , coef21 =
        ↪  sum(np.log(np.array(df))*(b1-1),sum((np.array(df)/a1)**b1)

        return -(coef + coef11 - coef21)
```

```python

    for j in tqdm(range(4, n-4)):

        optim = minimize(lambda x: funct_2(df, j, x), Vect0, method="nelder-mead", tol=1e-10)
        t = funct_2(df,j,optim.x)
        if t<taille:
            taille = t
            k = j

    #la fonction à minimiser afin de trouver les parametres optimals
    optim1 = minimize(lambda x: funct_1(df[:k], x), Vect0[:2], method="nelder-mead", tol=1e-10)
    optim2 = minimize(lambda x: funct_1(df[k:], x), Vect0[2:], method="nelder-mead", tol=1e-10)

    #retourner les parametres à estimer a1,b1 ~ W1
    #a2,b2 ~ W2
    #ainsi le parametre k

    return {"a1": optim1.x[1],"b1": optim1.x[0],"a2":optim2.x[1],"b2": optim2.x[0], "k*": k}

"""## Les graphes de méthode analytique + Vrai semblance"""

# tracée du cours des differentes actions de l'indice
def cours_action(data,date,nom_action,k1,k2):
        #Changer ici l'indice
    import datetime
    start=datetime.datetime(2019,1,1)
    end=datetime.datetime.today()
    import matplotlib.pyplot as plt
    fig=plt.figure(figsize=(12,6))
    plt.gcf().subplots_adjust(bottom=0.25)
    ax=fig.add_subplot(111)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')
    ax.set_xlabel('Index')
    ax.set_ylabel('Adjusted closing price')

    plt.plot(data)

    #Ploter deux lignes verticale des K trouvés pour les deux modèles
    plt.vlines(k1,0, max(data)+4, colors='g',label = "k1 = "+str(k1))
    plt.legend()

```

```python
    plt.vlines(k2, 0, max(data)+4, colors='r', linestyles= 'dashed',label = "k2 = "+str(k2))
    plt.legend()


    plt.title('Cours de l\'action %s du %s/%s/%s au %s/%s/%s'%(nom_action,start.day,start.month,start.year,end.day,end.month,end.year))
    ax.yaxis.grid(True)
    ax.xaxis.grid(True)
    fig.autofmt_xdate() # Corriger le chevauchement
    plt.tight_layout()
    plt.show()
    plt.close()

for i in range(len(Liste_donnees)):
    print('---------------------------------------------')
    print('--------------------------------------------')
    F1 = First_Method_analytic(Y(data[i]))  #Ici on doit changer l'indice pour qu'on puisse tester sur tous les bases de données
    F2 = Second_Method_log_vrais(Y(data[i]))
    print(F1)
    print(F2)
    cours_action(data[i],data_time[i],Liste_donnees[i],F1['k*'],F2['k*'])
```

A.2 Test de Kolmogorov-Smirnov

```python
# -*- coding: utf-8 -*-

import os
Liste_donnees=os.listdir('CAC40/DonCAC40/')
# %matplotlib inline

#####  pour tracer les graphes des données

for i in range(len(Liste_donnees)):
    F='CAC40/DonCAC40/'+Liste_donnees[i]
    data = pd.read_csv(F, delimiter=',')
    data=data.dropna() # supprimer NA s'il y en a
    data=data[data['Volume']!=0] # supprimer les données qui n'ont pas de sens
    data=data[(data['Date'] >='2019-01-01')]
    data = data.reset_index()

    # sauvegarder dans un nouveau dossier "selected"
    df = data[['AdjClose', 'Date']]
    df.to_csv("selected/%s.csv"%Liste_donnees[i][:-4], index = False)

"""# Algo K-S test

```

```python
### Test homogéniété
"""

import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import scipy
import scipy.stats
import random as rd

# Fonction de répartition empirique en t
# x échantillon ordonné
# cdfx fonction de répartion empirique aux points x
# avec fct indicatrice {X<=t}
def ecdf(t,x,cdfx):
    if t<x[0]:
        return 0
    if t>=x[-1]:
        return 1
    i = 0;
    while t>=x[i]:
        i=i+1
    return cdfx[i-1]

# Calcul des distances
# entre fct de repartition de x et fct de répartition de y
# aux points Z (avec Z les valeurs de x et de y ordonnées)
def distances(Z,x,cdfx,y,cdfy):
    D=np.zeros(len(Z))
    j=0;
    for i in Z:
        D[j] = abs(ecdf(i,x,cdfx)-ecdf(i,y,cdfy))
        j=j+1
    return D;

# Test de Kolmogorov-Smirnov d'homogénéité entre deux distributions
# On utilise l'approximation de l'article de Press et Teukolsky
# On calcule la p-value à une précision eps
def TEST_KS(X,Y,eps = 10**(-6)):
    ## Calcul de la statistique de test
    # Tri des données
    x=sorted(X)
    y=sorted(Y)
    n=len(x)
    m=len(y)

    # Calcul des fonctions de répartition empiriques
    cdfx = np.arange(1,n+1)/float(n)
    cdfy = np.arange(1,m+1)/float(m)
```

```python

    # On rassemble les données dans Z et on les trie
    Z=sorted(np.concatenate((X,Y),axis = None))

    # Calcul des distances entre fct de répartition empiriques
    D = distances(Z,x,cdfx,y,cdfy)
    # On garde la plus grande distance
    # C'est la statistique de test
    Dmax = max(D)

    ## Approximation de la p-value
    k= np.sqrt(n*m/(n+m))
    # Calcul du nombre de termes à sommer pour une précision de eps
    N = np.sqrt(np.log(eps)/(-2*(Dmax*k)**2))
    i = np.arange(1,N)
    # Calcul de l'approximation de la p-value
    p = sum(-2*((-1)**i)*np.exp(-(2*i**2)*((Dmax*k)**2)))

    # # On renvoie Dmax et pvalue
    # return 'my_KS_Result(statistic=',Dmax, 'pvalue=', p
    return  p

# Algo test KS qui prend en compte de l'ensemble de données et le point de
↪  rupture
def my_KS(Z,K):
    X = Z[np.arange(0,K)]
    Y = Z[np.arange(K,len(Z))]
    return TEST_KS(X,Y)

"""## Exemples d'Applications"""

# Exemple test avec des petits échantillons
X=np.array([13,4,5,17,6,3,1,15])
Y=np.array([2,7,8,9,10,6.5])
TEST_KS(X,Y)

Z=np.concatenate((X,Y),axis=None)
K=len(X)
my_KS(Z,K) # même valeur que TEST_KS(X,Y), donc c'est bon

scipy.stats.ks_2samp(X,Y)
# Normal que ce ne soit pas la même p-value.
# notre p-value est approchée pour des n et m grands (m,n tailles
↪  d'échantillons des deux cotés de la rupture)
# Python utilise la loi exacte quand n et m petits



# Deux échantillons relativement grands
```

```
X = scipy.stats.norm.rvs(size=100, loc=0., scale=1)
Y = scipy.stats.norm.rvs(size=200, loc=0.5, scale=1)
Z=np.concatenate((X,Y),axis=None)
K=len(X)
my_KS(Z,K)

scipy.stats.ks_2samp(X,Y)

my_KS(Z,298) # trouver le k qui maximise "statistics"

"""# Les ruptures trouvées"""

k_maxim = [300 , 292 , 292 , 292 , 287 , 292 , 292 , 296,
            403 , 302 , 290 , 292 , 329 , 266 , 476 , 295,
           357 , 480 , 5 , 472 , 368 , 365 , 478 , 368,
            211 , 285 , 292 , 475 , 486 , 300 ,477 ,581,
           516 , 371 , 292 , 295 , 542 , 486 ]
len(k_maxim)

k_anlyt = [ 302 , 300 , 292 , 298 , 284 , 203 , 302 , 271
, 402 , 302 , 301 , 298 , 328 , 266 , 476 , 301
, 357 , 480 , 5 , 341 , 372 , 364 , 477 , 364
, 211 , 285 , 491 , 489 , 302 , 305 , 477 , 587
, 300 , 290 , 301 , 295 , 687 , 340 ]
len(k_anlyt)

filenames =
↪  ["ACA.PA.csv","AC.PA.csv","AI.PA.csv","AIR.PA.csv","ALO.PA.csv","BN.PA.csv",

         ↪  "BNP.PA.csv","CA.PA.csv","CAP.PA.csv","COFA.PA.csv","CS.PA.csv","DGM.PA.csv",

         ↪  "DG.PA.csv","EDF.PA.csv","EN.PA.csv","GLE.PA.csv","HCMLF.csv","HO.PA.csv",

         ↪  "IDL.PA.csv","IPN.PA.csv","KER.PA.csv","LR.PA.csv","MC.PA.csv","ML.PA.csv",

         ↪  "ORA.PA.csv","OR.PA.csv","PUB.PA.csv","RI.PA.csv","RNO.PA.csv","SAF.PA.csv",

         ↪  "SAN.PA.csv","SEV.PA.csv","SGO.PA.csv","SU.PA.csv","TTE.PA.csv","VIE.PA.csv",
         "VIV.PA.csv","VK.PA.csv"]
len(filenames)

"""# Réalisation du test K-S sur les ruptures trouvés"""

def py_KS(Z,K):
    X = Z[np.arange(0,K)]
    Y = Z[np.arange(K,len(Z))]
    return scipy.stats.ks_2samp(X,Y)

"""## Test du modèle max de vraisemblance
```

```

"""

for i in range(len(filenames)):
# for i in range(2):
    F='selected/'+filenames[i]
    data = pd.read_csv(F, delimiter=',')

    print(my_KS(data['AdjClose'],k_maxim[i]))
    # print(py_KS(data['AdjClose'],k_maxim[i]))

"""## Test du modèle analytique

"""

# test analytique

for i in range(len(filenames)):
# for i in range(2):
    F='selected/'+filenames[i]
    data = pd.read_csv(F, delimiter=',')

    print(my_KS(data['AdjClose'],k_anlyt[i]))
    # print(py_KS(data['AdjClose'],k_anlyt[i]))
```