Mémoire de recherche

valant évaluation dans le cadre de :

Diplôme : Master de statistiques pour l'évaluation et la prévision, 1ère année
Année universitaire : 2022-2023
Module d'enseignement : SEP0851
Responsable : Philippe Regnault
Comptant pour : 50 %

Méthodes de détection de rupture offline

Elif ERTAS elif.ertas@etudiant.univ-reims.fr

Métriques

Finalisé le : 22 mai 2023
Page(s) : 23
Référence(s) : 0
Figure(s) : 8
Table(s) : 0
Théorème(s) : 0

Résumé : Le mémoire de recherche examine les méthodes de détection de rupture offline, qui sont utilisées pour identifier les changements significatifs dans les données. On étudiera les algorithmes d'apprentissage automatique, en plus de l'aspect mathématique de ces derniers. L'objectif est de comprendre les avantages et les limites de chaque méthode pour aider à sélectionner la meilleure approche en fonction du contexte de la recherche.

Mots-clés : Analyse de signal, Ruptures de tendances, Prédiction de la détection de ruptures.

Matériel supplémentaire :
git: https://github.com/pregnault/urcadown
: https://search.r-project.org/CRAN/refmans/changepoint/html/wave.c44137.html

Table des matières

Remerciements

Je tiens tout d'abord à remercier mon directeur de recherche, M^r Regnault, pour son soutien, ses conseils et ses encouragements tout au long de ce projet de recherche. Son expertise, sa disponibilité et sa patience ont été inestimables et ont grandement contribué à la réussite de ce mémoire.

Je souhaite également remercier M^r Maës, pour son intérêt et le temps consacré à ma soutenance.

Je remercie aussi les professeurs et l'ensemble de l'équipe pédagogique du master de statistique pour l'évaluation et la prévision, pour leurs enseignements durant cette première année qui m'auront notamment permis d'aboutir à ce mémoire.

Je remercie aussi particulièrement M^{me} Gautherat qui a été présente tout au long de cette première année de master, et de m'avoir apporté des conseils pour le bon déroulement de ce mémoire de recherche.

Enfin, je tiens à remercier mes proches, ma famille et mes amis, pour leur soutien et leur encouragement tout au long de ce mémoire de recherche.

Je suis reconnaissante envers toutes les personnes qui ont contribué à la réalisation de ce travail, et je tiens à leur exprimer toute ma gratitude pour leur aide précieuse.

1. Introduction

Sous la dénomination « détection de rupture » on regroupe un ensemble de méthodes statistiques visant à estimer les instants d'un processus temporel qui change brutalement. On distingue les méthodes onlines et offlines.

Les méthodes onlines permettent de faire la détection en temps réel, c'est-à-dire pendant l'enregistrement du signal. Tandis que les méthodes offlines font la détection à postériori. Étant donné que ce sont deux grosses parties indépendantes, par les conseils de mon tuteur M. Regnault j'ai décidé d'effectuer mes recherches uniquement pour les méthodes offlines.

L'intérêt de ces algorithmes de détection de rupture offline est donc d'identifier le découpage d'un signal enregistré au préalable. Tous les algorithmes font l'hypothèse que les séries temporelles étudiées sont des concaténations de séries possédantes chacune une dynamique propre, décrite par un modèle. Les algorithmes de détection sont caractérisés par trois éléments : une fonction de coût, une méthode de recherche et une contrainte sur le nombre de changements.

Dans le cadre de mon mémoire de recherche, j'ai décidé d'étudier la provenance de l'algorithme PELT et de comprendre son fonctionnement plus en détail.

2. Détection de rupture offline

Une tâche courante dans le traitement du signal est l'identification et l'analyse de systèmes complexes dont l'état change une ou plusieurs fois. C'est le cas pour des systèmes industriels (surveillance industrielle), des phénomènes physiques (la pousse des arbres) ou l'activité humaine (activité électrique du cerveau). Nous sommes dans des cas où les données sont surveillées en permanence par des capteurs pour pouvoir ensuite y extraire le signal qui sera étudié par la suite. Cela se nomme la détection offline car on cherche les changements a posteriori, après avoir enregistré le signal.

Quel est le but de la détection de rupture ? Les premiers travaux concernant la détection de rupture se déroulent vers les années 1950, dont le but était de localiser un changement de la moyenne. Concrètement, on cherche à trouver des changements dans le modèle d'un signal ou d'une série temporelle.

Exemple : L'analyse de la marche d'une personne illustrée sur l'organigramme de la figure 1 faite dans l'article [TOV20] [1].

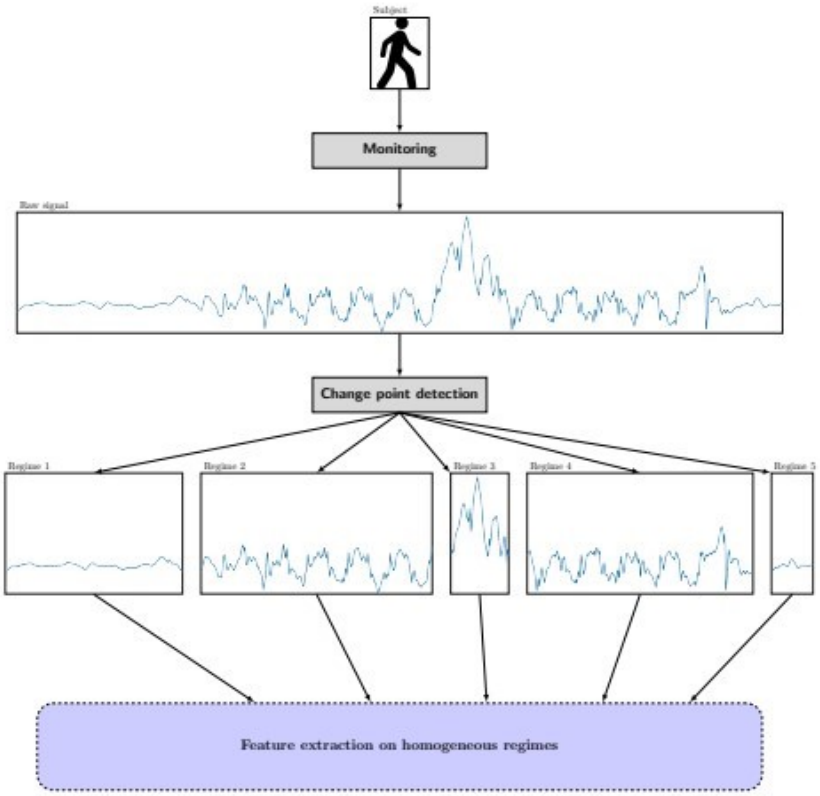


FIGURE 1 – Ornigramme pour l'analyse de la marche.
Cohorte : Une personne aux corpulences dans la moyenne et en bonne santé.
Lecture : On observe que le signal initial est décomposé en 5 régimes.

Dans cette analyse, les mouvements du patient sont surveillés avec des accéléromètres et des gyroscopes lors d'activités simples comme marcher et courir, mais à des vitesses différentes. On observe alors 5 régimes différents qui sont découpés du signal initial enregistré. En effet, il y a bien 5 phases différentes pendant lesquelles le signal change légèrement comme les régimes 1 et 5 ou complètement comme les régimes 3 et 4.

1. http://arxiv.org/abs/1801.00718

Tous les signaux, quel que soit le domaine d'analyse sont très différents et peuvent parfois nécessiter de méthodes différentes pour procéder à une étude de détection de rupture. Il existe une multitude de méthodes, qui sont des algorithmes, dont certains sont expliqués et testés dans l'article [TOV20]. Chaque algorithme est défini par une fonction de coût, une méthode de recherche et une contrainte :

— La fonction de coût est un score, choisi en fonction de la modélisation retenue que l'on cherche à minimiser pour trouver les instants de changements de dynamique.
— La méthode de recherche est la méthodologie employée pour minimiser la fonction de coût. Il y a deux familles : méthodes exactes (couteuses en temps de calcul) et les méthodes approchées.
— La contrainte est le nombre de changements : le nombre de changements est-il connu a priori ou non. En général non, donc il est nécessaire d'estimer ce nombre en ajoutant une pénalité à la fonction de coût croissante avec le nombre de changements.

On distingue les algorithmes en deux catégories en fonction que l'on sait ou non son nombre de changements à l'avance. On y trouve en particulier l'algorithme PELT qui fonctionne pour les signaux où on ne connaît pas forcément le nombre de changements à l'avance et sur lequel nous allons nous intéresser.

Dans la suite, $(x_t)_{t=0,\dots,T}$ désigne la réalisation des $T+1$ premières coordonnées d'une suite de variables aléatoires réelles $(X_t)_{t=0,\dots,T}$ telle qu'il existe une partition $\mathcal{T}^{\text{true}} = \{(0 = \tau_0, \dots, \tau_1), (\tau_1 + 1, \dots, \tau_2), \dots, (\tau_K + 1, \dots, \tau_{K+1} = T)\}$ telle que sur chaque intervalle $I_k = \tau_k + 1, \dots, \tau_k$, $k = 0, \dots, K$, les variables X_t, $t \in I_k$ ont même loi et sur deux intervalles consécutifs, les lois respectives sont distinctes.

Exemple 1 (Changement de moyennes dans une suite de variables indépendantes gaussiennes)**.** —

En guise d'exemple fil-rouge (développé tout au long des deux prochaines parties), on considère un vecteur $(X_t)_{t\in\{0,\dots,T\}}$ de variables indépendantes telles que :

— $\forall t \in (0, ..., t_1), X_t \sim \mathcal{N}(m_0, \sigma)$
— $\forall t \in I_k, X_t \sim \mathcal{N}(m_k, \sigma)$, $k \in \{1, \dots, K\}$,

où $\sigma > 0$, $(x_k)_{k=1,\dots,N}$ est une suite d'entiers tel que $\forall k$, $m_k \neq m_{k+1}$ [2].

Les méthodes de détection de rupture servent à estimer les entiers t_k et les comportements en loi sur chaque intervalle à partir des observations $(x_t)_{t=0,\dots,T}$ Précisément, on cherche à construire une partition $\mathcal{T}$ de $(0, ..., T)$ de K+1 intervalles :

$$T = \{(0 = t_0, \dots, t_1), (t_1 + 1, \dots, t_2), ..., (t_K + 1, \dots, t_{K+1} = T)\}$$

Telle que le comportement en loi de $(x_t)_t \in (t_k + 1, \dots, t_{k+1})$ soit constant.

2. C'est-à-dire que l'on peut avoir deux moyennes identiques dans l'ensemble des intervalles, mais pas sur deux intervalles consécutifs.

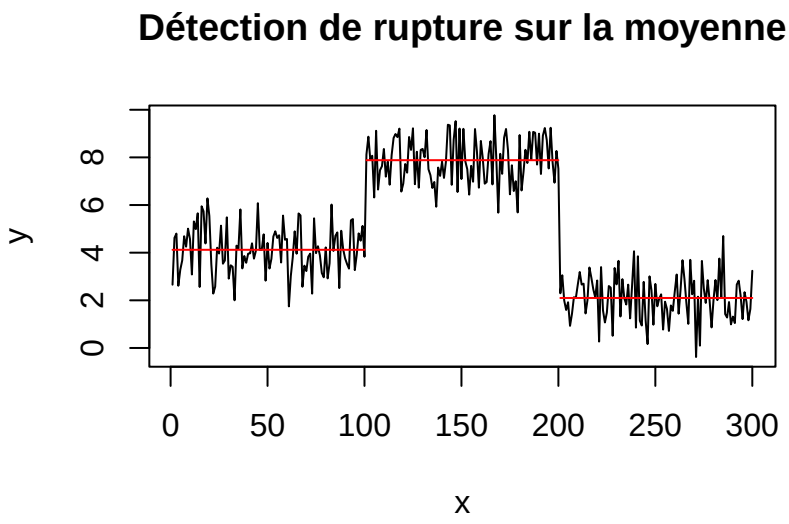


FIGURE 2 – Exemple de détection de rupture en moyenne.
Cohorte : Trois échantillons suivant une loi normale de même écart-type mais de moyenne volontairement différente.
Lecture : On observe une rupture de moyenne entre chaque échantillon, par exemple le premier échantillon suit une loi normale de moyenne 4 tandis que le deuxième échantillon suit une loi normale de moyenne 8.

La figure 2 illustre la trajectoire (simulée) d'un tel vecteur pour $K = 2$, $\tau_1 = 100$, $\tau_2 = 200$, $\tau_3 = T = 300$, $\sigma = 1$ et $m_1 = 4$, $m_2 = 8$, $m_3 = 2$. Les échantillons étant simulés à notre demande, on savait déjà qu'il y aurait 2 ruptures et où elles seraient situées. On observe bien visuellement que l'algorithme de détection de rupture sur la moyenne a bien découpé le signal aux bons endroits.

Dans cet exemple, on n'avait pas besoin de renseigner le nombre de ruptures à l'avance pour faire la détection. Cependant, ce n'est pas toujours le cas. Les algorithmes de détection de rupture se dissocient en 2 catégories, certains ont besoin de savoir à l'avance combien il y aura d'intervalles d'autres non.

2.1. Estimation de la partition si K est connu.

Supposons $K << T$ connu [3] .

Pour construire une partition la plus proche possible de la partition théorique inconnue $\mathcal{T}^{\text{true}}$, ainsi que le changement de régime entre ces intervalles, on introduit une fonction de coût qui à chaque vecteur d'observations $(x_t)_{t=0,\dots,T}$ et chaque partition $\mathcal{T}$ associe un réel positif $c((x_t)_{t=0,\dots,T}, \mathcal{T})$ qui est d'autant plus petit que la partition T est proche de la partition théorique. Ainsi, la partition estimée à partir des observations $(x_t)_{t=0,\dots,T}$ [4] sera

$$\mathcal{T}^* = \text{Argmin}_{\mathcal{T}} \ \ c(x_{0:T}, \mathcal{T}).$$

Un choix usuel consiste à utiliser l'opposé de la log-vraisemblance de l'échantillon, lorsque celle-ci est connue.

3. $K << T$ signifie un nombre d'intervalle très petit par rapport à T.

4. $(x_t)_{t=0,\dots,T}$ est une notation qui signifie la même chose que $x_{0:T}$.

Exemple 2 (Cas gaussien iid, suite)**.** — Sous les hypothèses de l'exemple 1 on est amené à minimiser en $\mathcal{T}$ l'erreur quadratique

$$c((x_t)_{t=0,\dots,T}, T) = \sum_{k=1}^{K} \sum_{t\in(t_{k-1},\dots,t_k-1)} (x_t - \bar{x}_t)^2 \qquad o\grave{u} \quad \bar{x}_t = \frac{1}{I_k} \sum_{t\in I_k} x_t.$$

La partition estimée est donc

$$T^* = \text{Argmin}_{\mathcal{T}} \sum_{k=1}^{K} \sum_{t\in(t_{k-1},\dots,t_k-1)} (x_t - \bar{x}_t)^2$$

On est ainsi amené à résoudre un problème d'optimisation complexe, car l'ensemble des partitions $\mathcal{T}$ possibles est très grand.
On a alors deux méthodes :

— Les méthodes exactes : on détermine effectivement T^*
— Les méthodes approchées : on estime T^*

2.2. Estimation de la partition lorsque K est inconnu

Si K est inconnu, on pénalise la fonction de coût en y ajoutant une pénalité qui s'exprime comme une fonction croissante du nombre K de parties.

$$\min_{T} \sum_{k=1}^{K} \sum_{t\in(t_{k-1},\dots,t_k-1)} (x_t - \bar{x}_t)^2 + \beta \text{ pen}(K)$$

Quelques exemples de pénalités classiques incluent les suivantes :

— AIC : pen$(K) = K$ et $\beta > 0$ constante ;
— BIC : pen$(K) = K$ et $\beta = b \cdot log(n)$
— Lavielle : $pen(K) = K$; et β adaptif, c'est-à-dire, fonction des observations ;
— Birgé et Massart : $pen(K) = \beta_1 K + \beta_2 \log K$, avec β_1, β_2 adaptatifs.

Dans la suite, on s'intéressera uniquement aux pénalités additives, c'est-à-dire, telles que la pénalité de la partition soit la somme des pénalités de chaque partie. Par exemple, les pénalités linéaires AIC, BIC ou de Lavielle.

3. Algorithmique

L'algorithme de détection de rupture PELT [5] observe les changements de la moyenne dans une série temporelle. Il s'agit d'une méthode de segmentation qui cherche à identifier les points de rupture dans une série temporelle en minimisant un critère de coût basé sur la somme des carrés des écarts à la moyenne. Plus précisément, l'algorithme PELT cherche à diviser la série temporelle en sous-séries de manière à minimiser le critère de coût tout en imposant une contrainte, c'est-à-dire en limitant le nombre de points de rupture. Ainsi, l'algorithme identifie les changements significatifs dans la moyenne de la série temporelle.

L'algorithme PELT est dérivé de l'algorithme de programmation dynamique [6], lui-même dérivé d'un algorithme dit "du voyageur de commerce [7]". [8]

En outre, PELT fournit une solution exacte, ce qui signifie que les points de rupture détectés sont les mêmes que ceux qui seraient détectés par une recherche exhaustive qui elle a une complexité de l'ordre de $\mathcal{O}(K^n)$. D'un autre côté, l'algorithme de programmation dynamique peut être utilisé pour résoudre une grande variété de problèmes algorithmiques différents. Cependant, sa complexité temporelle peut être polynomiale, ce qui signifie qu'il peut être très lent pour des entrées plus grandes.

3.1. Programmation dynamique

L'algorithme dynamique est une méthode pour résoudre des problèmes algorithmiques en utilisant des résultats déjà calculés pour résoudre des sous-problèmes ultérieurs plus rapidement. C'est-à-dire que les résultats ultérieurs dépendent forcément du premier résultat que l'on calcule. Dans une série temporelle, ça peut être le premier point ou le dernier.

L'algorithme de programmation dynamique peut être utilisé pour résoudre une grande variété de problèmes algorithmiques différents. Cependant, sa complexité temporelle peut être polynomiale, ce qui signifie qu'il peut être très lent pour des entrées plus grandes. Il a une complexité [9] de $\mathcal{O}(Kn^2)$ voire $\mathcal{O}(Kn^3)$ lorsque K est de l'ordre de n.

Optimal Partitioning

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
A measure of fit $\mathcal{C}(\cdot)$ dependent on the data.
A penalty constant β which does not depend on the number or location of changepoints.

Initialise: Let n = length of data and set $F(0) = -\beta$, $cp(0) = NULL$.

Iterate for $\tau^* = 1, \ldots, n$

1. Calculate $F(\tau^*) = \min_{0 \le \tau < \tau^*} \left[F(\tau) + \mathcal{C}(y_{(\tau+1):\tau^*}) + \beta \right]$.
2. Let $\tau' = \arg \left\{ \min_{0 \le \tau < \tau^*} \left[F(\tau) + \mathcal{C}(y_{(\tau+1):\tau^*}) + \beta \right] \right\}$.
3. Set $cp(\tau^*) = (cp(\tau'), \tau')$.

Output the change points recorded in $cp(n)$.

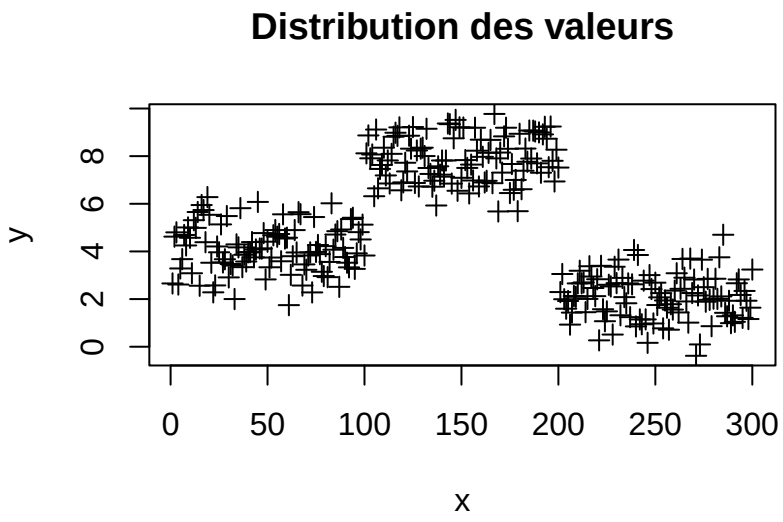
FIGURE 3 – Description formelle de l'algorithme dynamique.
Source : Algorithme issue de l'article [KFE12]

5. "Pruned Exact Linear Time" qui peut être traduit en français par "Temps Linéaire Exact Élagué".
6. Algorithme proposé principalement par Jackson en 2005.
7. Algorithme proposé par Auger et Lawrence en 1989, plus d'information dans l'article [AL89].
8. Plus d'informations sur ces algorithmes et leur provenance dans l'article de Killick, Fearnhead et Eckley [KFE12].
9. K est le nombre de segments dans la série et n est la longueur de la série.

On observe que pour utiliser l'algorithme dynamique, il faut renseigner la série, la fonction de coût que l'on veut appliquer et la pénalité β.

L'algorithme commence par initialiser $F(0) = -\beta$ qui est la valeur optimale de la somme de la fonction de coût et $cp(0) = NULL$ qui est la liste des points de changement.
Ensuite pour chaque instant $cp(\mathcal{T}^*)$ allant de 1 à n, l'algorithme calcule la valeur optimale de $F(\mathcal{T}^*)$. Cette valeur est calculée en minimisant la somme de la valeur précédente de $F(\mathcal{T})$ et le coût de la partie de la série temporelle entre $\mathcal{T}+1$ et $\mathcal{T}^*$, ajouté à la constante de pénalité β.
L'algorithme enregistre ensuite le point $\mathcal{T}'$ qui minimise cette expression. Puis enregistre la liste des points de changement $cp(\mathcal{T}^*)$ jusqu'à ce point, en incluant $\mathcal{T}'$. Finalement, l'algorithme retourne la liste des points de changement enregistrés dans $cp(\mathcal{T}^*)$.

Appliquons l'algorithme étape par étape sur notre exemple de départ :



Initialisation : $n = 300$, $F(0) = -\beta$ et $cp(0) = NULL$.

1er tour de boucle : $\mathcal{T}^* = 1$

1. $F(\mathcal{T}^*) = F(1) = \min_{0 \leq \mathcal{T} < 1} \{F(\mathcal{T}) + c(x_{1:1}) + \beta\} = F(0) + 0 + \beta = -\beta + \beta = 0$
2. $\mathcal{T}' = 0$
3. $cp(1) = \begin{cases} cp(\mathcal{T}'), & \text{si } \mathcal{T}' \neq 0 \\ cp(\mathcal{T}', \text{NULL}), & \text{si } \mathcal{T}' = 0 \end{cases} = cp(\mathcal{T}', \text{NULL})$

2ème tour de boucle : $\mathcal{T}^* = 2$

1. $F(\mathcal{T}^*) = F(2)$
   $= \min_{0 \leq \mathcal{T} < 2} \{F(\mathcal{T}) + c(x_{\mathcal{T}+1:2}) + \beta\}$
   $= \min_{0 \leq \mathcal{T} < 2} \{F(0) + c(x_{1:2}) + \beta; F(1) + c(x_{2:2}) + \beta\}$
   $= \min_{0 \leq \mathcal{T} < 2} \{c(x_{1:2}); \beta\}$
2. $\mathcal{T}' = \begin{cases} 0 \text{ si } c(x_{1:2}) < \beta \\ 1 \text{ si } c(x_{1:2}) \geq \beta \end{cases}$
3. $cp(2) = \begin{cases} cp(\mathcal{T}'), & \text{si } \mathcal{T}' \neq 0 \\ cp(\mathcal{T}', \text{NULL}), & \text{si } \mathcal{T}' = 0 \end{cases}$

k ème tour de boucle : On regarde $\forall \mathcal{T}^* < k$ quelle est la meilleure partition. C'est-à-dire que pour calculer F(k), on regarde d'abord pour $\mathcal{T} = 0$ c'est-à-dire qu'il n'y a pas de point de changement, puis pour $\mathcal{T} = 1$ qui correspond à deux partitions : le point 1 tout seul et la partition de 2 à k.
Jusqu'à $\mathcal{T} = l$ qui correspond aux partitions de 1 à l (la meilleure partition des l premiers éléments) et de l+1 jusqu'à k. Tout cela jusqu'à $\mathcal{T} = k - 1$ où l'on aura déjà calculer à l'itération précédente $F(k-1)$.

Puis on applique cela jusqu'à n = 300.

On observe donc bien l'amélioration de l'algorithme, car on a déjà calculé à l'itération précédente la meilleure partition des éléments précédents et donc on a une complexité algorithmique meilleure.

3.2. Algorithme PELT

L'ajout du terme "élagué" dans le nom fait référence à la possibilité de réduire ou d'éliminer certaines parties du traitement afin d'améliorer l'efficacité de l'algorithme, tout en maintenant la garantie d'un résultat exact. En effet, l'algorithme PELT est issu de l'algorithme de programmation dynamique, mais on ne calcule pas pour chaque intervalle possible uniquement les intervalles où il y a un coût assez important pour qu'on le prenne en compte. Cela permet d'avoir moins de calcul lors de l'application de l'algorithme. L'algorithme PELT a été introduit pour trouver la solution exacte de ce problème lorsque la pénalité est linéaire.
Le nombre de changements étant inconnu, la détection de points de rupture consiste à résoudre le problème d'optimisation discret suivant :

$$\min_{T} V(T) + pen(T)$$

où $pen(T)$ est une mesure approximative de la complexité de la segmentation de T et $V(T)$ correspond à $V(T, y) := \sum_{k=0}^{K} c(y_{t_k \ldots t_{k+1}})$ où $c(.)$ est la fonction de coût.

La méthode de recherche de cet algorithme est optimale, elle permet de trouver une solution exacte du problème.

La principale motivation de l'algorithme de PELT est de réduire la complexité en temps à une complexité linéaire. Cet algorithme est optimisé pour avoir une complexité linéaire en temps, ce qui signifie qu'il peut traiter une série temporelle très rapidement, même si elle est très longue. Cet algorithme a une complexité de l'ordre de $\mathcal{O}(Kn)$.

PELT Method

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
A measure of fit $\mathcal{C}(.)$ dependent on the data.
A penalty constant β which does not depend on the number or location of changepoints.
A constant K that satisfies equation 4.

Initialise: Let n = length of data and set $F(0) = -\beta$, $cp(0) = NULL$, $R_1 = \{0\}$.

Iterate for $\tau^* = 1, \ldots, n$

1. Calculate $F(\tau^*) = \min_{\tau \in R_{\tau^*}} \left[F(\tau) + \mathcal{C}(y_{(\tau+1):\tau^*}) + \beta\right]$.
2. Let $\tau^1 = \arg\left\{\min_{\tau \in R_{\tau^*}} \left[F(\tau) + \mathcal{C}(y_{(\tau+1):\tau^*}) + \beta\right]\right\}$.
3. Set $cp(\tau^*) = [cp(\tau^1), \tau^1]$.
4. Set $R_{\tau^*+1} = \{\tau \in R_{\tau^*} \cup \{\tau^*\} : F(\tau) + \mathcal{C}(y_{\tau+1:\tau^*}) + K \leq F(\tau^*)\}$.

Output the change points recorded in $cp(n)$.

FIGURE 4 – Description formelle de l'algorithme PELT.
Source : Algorithme issue de l'article [KFE12]

On observe que pour utiliser l'algorithme PELT, il faut renseigner la série, la fonction de coût que l'on veut appliquer, la pénalité β et une constante $K \geq 0$.
Cette constante K est introduite pour choisir si un point doit être pris en compte en tant que point de changement ou non. Elle doit satisfaire l'équation suivant :

$$c(y_{(t+1):s}) + c(y_{(s+1):T}) + K \leq c(y_{(t+1):T})$$

L'équation compare le coût total de trois parties d'une séquence de données, notée "y". La séquence est divisée en trois parties distinctes : de l'indice "t+1" à "s", de "s+1" à "T" et de "t+1" à "T".

Le côté gauche de l'équation est la somme des coûts de la première partie de la séquence (de "t+1" à "s") et de la deuxième partie (de "s+1" à "T"), à laquelle on ajoute une constante K. Le côté droit de l'équation est le coût de la séquence entière, de "t+1" à "T".

L'inégalité indique que le coût total des deux premières parties de la séquence, plus la constante K, est inférieur ou égal au coût total de la séquence entière.

En d'autres termes, cette équation exprime une condition qui doit être satisfaite pour que la division de la séquence en deux parties soit justifiée. Si le coût total des parties individuelles, plus la constante K, est inférieur ou égal au coût de la séquence complète, alors cette division est considérée comme favorable et donc on prend en compte le point de changement.

Voici par un exemple la vérification de la condition en utilisant comme fonction de coût la norme euclidienne.

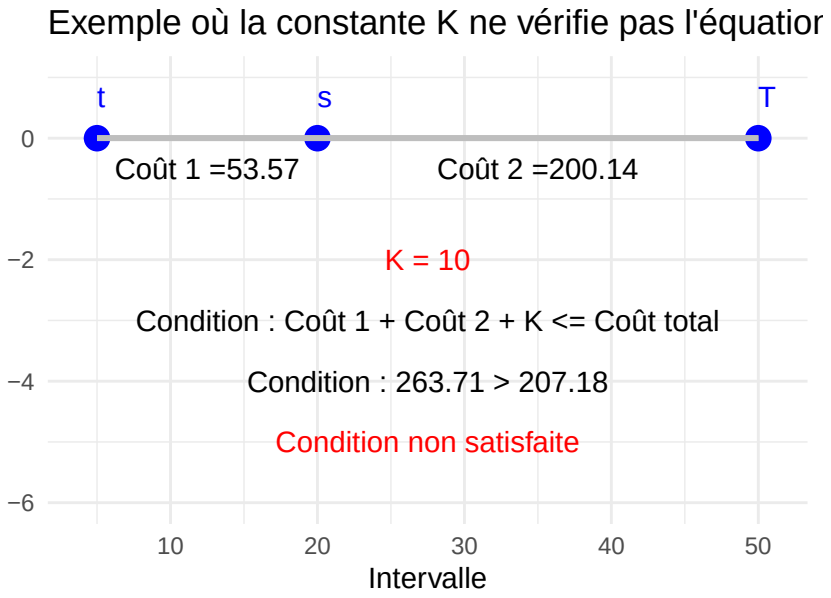


FIGURE 5 – Exemple où la constante K ne vérifie pas l'équation.
Lecture : On observe que la somme du coût du premier intervalle, du second intervalle et de la constante K est supérieure au coût de l'intervalle total. La condition n'est donc pas vérifiée, alors on ne prend pas le point s comme point de changement.

Maintenant en gardant les mêmes paramètres et en changeant uniquement la fonction de coût, on peut avoir un exemple où K vérifie la condition. Prenons comme fonction de coût par exemple :

$$c(y_I) = \begin{cases} 2s, & \text{si } I = (t+1:s) \\ 3(T-s), & \text{si } I = (s+1:T) \\ 4T, & \text{si } I = (t+1:T) \end{cases}$$

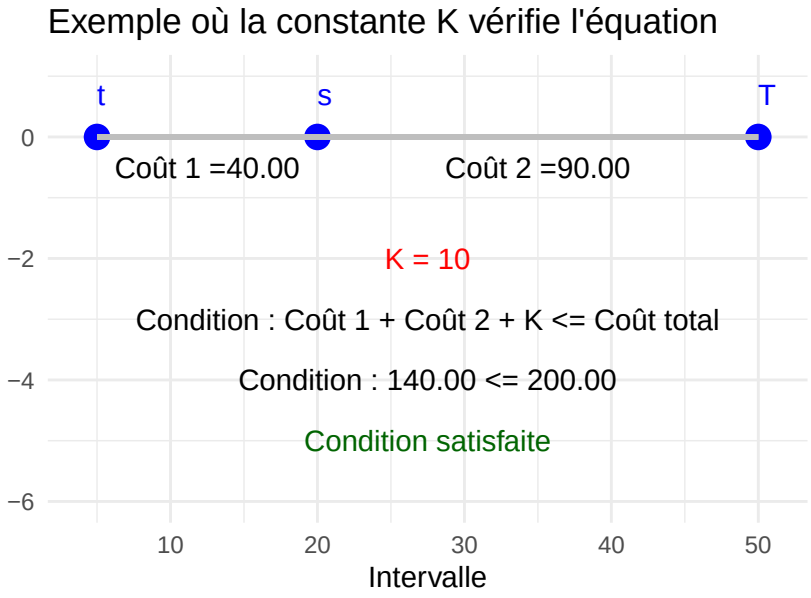


FIGURE 6 – Exemple où la constante K vérifie l'équation.
Lecture : On observe que la somme du coût du premier intervalle, du second intervalle et de la constante K est inférieure au coût de l'intervalle total. La condition est donc vérifiée, alors on prend en compte le point s comme point de changement.

Une fois les arguments renseignés, on peut alors faire dérouler l'algorithme.

L'algorithme commence par initialiser $F(0) = -\beta$ qui est la valeur optimale de la somme de la fonction de coût, $cp(0) = NULL$ qui est la liste des points de changement et $R_1 = 0$ ce qui signifie que le premier point de changement possible est à la position 0.

Ensuite pour chaque instant $cp(\mathcal{T}^*)$ allant de 1 à n, on calcule la meilleure qualité d'ajustement possible $F(\mathcal{T}^*)$ pour les données jusqu'à $cp(\mathcal{T}^*)$ en explorant les points de changement possibles dans $R_{\mathcal{T}^*}$, qui contient les indices des points de changement potentiels jusqu'à $\mathcal{T}^*$.

On détermine le premier point de changement $\mathcal{T}^1$ dans $R_{\mathcal{T}^*}$ qui minimise la qualité de l'ajustement. On ajoute le point de changement $\mathcal{T}^1$ à la liste $cp(\mathcal{T}^*)$.
Enfin, on calcule la nouvelle liste de points de changement potentiels $R_{\mathcal{T}^*+1}$ qui contient tous les points de changement dans $R_{\mathcal{T}^*}$ qui satisfont l'équation $F(\mathcal{T}) + c(y_{(T+1):T^*}) + K \leq F(\mathcal{T}^*)$.

Finalement, l'algorithme retourne la liste des points de changement enregistrés dans $cp(n)$.

4. Détection de rupture sur la hauteur des vagues

4.1. Explication des hauteurs des vagues

Les vagues sont des ondulations qui se propagent à la surface de l'eau. Elles sont créées par le vent, les marées, les courants océaniques et les séismes. La hauteur des vagues dépend de plusieurs facteurs, notamment la vitesse et la direction du vent, la profondeur de l'eau, la longueur d'onde de la vague et la force de la gravité.

Les vagues peuvent varier en hauteur, allant de quelques centimètres à plus de 30 mètres comme expliqué dans [OT] [10] . Les petites vagues qui se forment près du rivage sont généralement moins hautes que les vagues qui se forment en eau profonde. Les vagues peuvent également être classées en fonction de leur taille, avec des catégories allant de vagues "légères" à "énormes". La hauteur des vagues est souvent mesurée en utilisant la différence entre la crête et le creux de la vague, appelée l'amplitude de la vague. Elle peut être mesurée de différentes manières. Les scientifiques utilisent des capteurs spéciaux appelés bouées pour mesurer la hauteur des vagues. D'après [Wik23] [11] ils peuvent également utiliser des images satellites pour estimer la hauteur des vagues.

Le cycle lunaire affecte les marées, qui sont les fluctuations régulières du niveau de l'eau à la surface de la mer. Les marées sont principalement causées par les forces gravitationnelles exercées par la Lune et le Soleil sur la Terre. Lorsque la Lune est pleine, c'est-à-dire lorsqu'elle est alignée avec le Soleil et la Terre, la force gravitationnelle exercée par la Lune et le Soleil sur la Terre est combinée, ce qui provoque des marées plus hautes, appelées marées de vive-eau. Pendant les phases de premier et de dernier quartier, c'est-à-dire quand la Lune forme un angle droit avec la Terre et le Soleil, les forces gravitationnelles sont moins fortes, ce qui entraîne des marées moins élevées, appelées marées de morte-eau.

Les marées influencent à leur tour la hauteur et la force des vagues. Les vagues qui se forment pendant les marées de vive-eau peuvent être plus grandes et plus puissantes que celles qui se forment pendant les marées de morte-eau. Les marées peuvent également affecter la direction et la forme des vagues, en fonction de la topographie des fonds marins et des caractéristiques côtières.
En résumé, le cycle lunaire et les marées qu'il génère ont un impact significatif sur les vagues, leur hauteur, leur force et leur direction.

4.2. Base de données sur les vagues

Pour appliquer l'algorithme PELT nous allons utiliser une base de données sur la hauteur des vagues. Dans la librairie changepoint dans R, nous pouvons trouver un vecteur de taille 63651 qui se nomme wave.c44137. Ce vecteur correspond à la hauteur des vagues exprimée en mètre, mesurée par la bouée c44137. Ces données ont été obtenues auprès de Pêches et Océans Canada qui est un ministère du gouvernement du Canada qui est responsable de la gestion des ressources marines du pays dont on peut retrouver les informations sur le site officiel [Gou19] [12] . Les données sont prises à intervalles horaires, c'est-à-dire par heure, de janvier 2005 à septembre 2012.

Le vecteur étant très grand, nous allons restreindre notre étude sur les 5 000 premières observations (qui correspond environ aux 8 premiers mois) :

10. http://www.culture-maritime.com/fr/text-mme4-cours2
11. https://fr.wikipedia.org/w/index.php?title=Vague&oldid=200831696
12. https://www.meds-sdmm.dfo-mpo.gc.ca/isdm-gdsi/waves-vagues/data-donnees/index-fra.asp

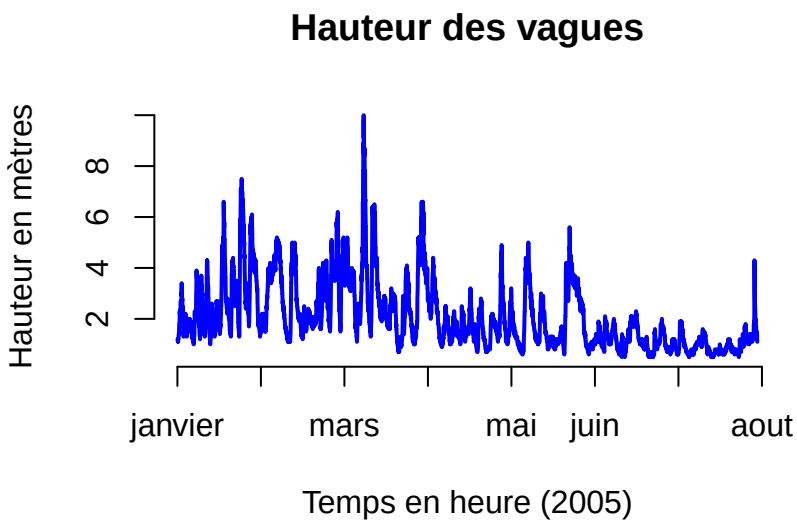


FIGURE 7 – Hauteur des vagues par heure.
Cohorte : Hauteur des vagues mesurée par la bouée c44137 de janvier 2005 à août 2005 prises à intervalles horaires.
Lecture : On observe qu'en début mars, il y a eu un pic de hauteur allant jusqu'à 10 mètres.

La hauteur des vagues n'est donc pas constante, elle oscille de façon sinusoïdale.

4.3. Détection de rupture en moyennes

Nous allons appliquer un algorithme de détection de rupture en moyenne par la méthode PELT.

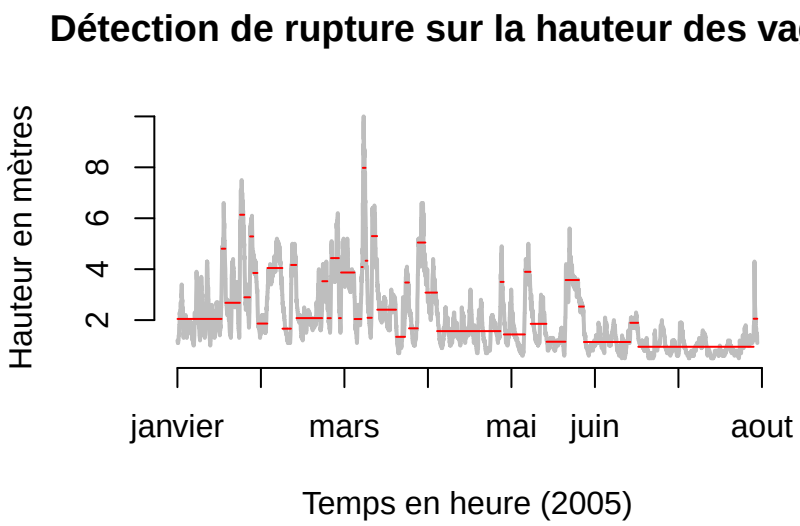


FIGURE 8 – Détection de rupture en moyenne des hauteurs des vagues.
Cohorte : Hauteur des vagues mesurée par la bouée c44137 de janvier 2005 à août 2005 prises à intervalles horaires.
Lecture : On observe que pendant tous le mois d'avril, on a une moyenne constante.

On observe qu'il y 40 (annexe 1) points de ruptures situés de façon non-régulière [13]. Il y a des périodes plus ou moins longues que d'autres. Cependant on peut dire qu'en hiver les moyennes des hauteurs des vagues ont tendance à varier davantage qu'en printemps et été.

Il est possible que la hauteur des vagues ait la même moyenne pendant les mois de juin et juillet en raison de plusieurs facteurs, tels que les conditions météorologiques, les changements saisonniers, ou les effets des courants océaniques.
Par exemple, les mois de juin et juillet peuvent avoir des conditions météorologiques similaires, telles qu'un même schéma de vent ou une même direction de houle, ce qui peut entraîner des hauteurs de vague similaires. De plus, les courants océaniques peuvent avoir un effet sur la hauteur des vagues, qui peut être similaire pendant ces deux mois.
Enfin, il peut y avoir des changements saisonniers qui affectent la hauteur des vagues, tels que la fréquence et la force des tempêtes, qui peuvent être similaires pendant les mois de juin et juillet.
Cependant, il est important de noter que ce ne sont que des hypothèses et que la hauteur des vagues peut varier d'une année à l'autre.

Pour conclure, le réchauffement climatique peut entraîner des variations dans les conditions météorologiques et donc affecter la hauteur des vagues. Des phénomènes tels que l'augmentation de la température de l'air et de l'eau, la modification des courants océaniques, la fonte des glaciers et l'élévation du niveau de la mer peuvent tous influencer la formation des vagues. Ces changements peuvent donc affecter la tendance des hauteurs de vagues observées au fil des ans.
Il est donc important de suivre les tendances et les variations des données sur une période plus longue pour déterminer si ces changements sont significatifs ou non.

13. Plus d'informations sur les positions dans l'annexe 2.

5. Conclusion

Dans le cadre de ce mémoire de recherche, je me suis intéressée aux méthodes de détection de ruptures offline, en mettant l'accent sur l'algorithme PELT. Mon objectif était de comprendre le fonctionnement de l'algorithme PELT et d'appliquer cette méthode à des données sur la hauteur des vagues pour détecter les points de rupture.

J'ai constaté que l'algorithme PELT offre une approche précise et efficace pour détecter les points de rupture dans une séquence de données. Son principal avantage réside dans sa capacité à fournir des résultats exacts tout en ayant une complexité linéaire, ce qui le rend applicable à des ensembles de données de grande taille.

La fonction de coût de l'algorithme PELT est très importante pour évaluer si une configuration de points de rupture est bonne ou non. Cette fonction de coût prend en compte deux choses : d'abord, elle regarde comment les points de rupture correspondent aux données localement, c'est-à-dire si les valeurs des données autour des points de rupture sont cohérentes. Ensuite, elle regarde combien de points de rupture il y a dans la configuration, en préférant les configurations avec moins de points de rupture.

En appliquant l'algorithme PELT aux données de la hauteur des vagues, j'ai pu observer son efficacité dans la détection des points de rupture significatifs. Les résultats obtenus nous ont permis de mettre en évidence les moments où des changements majeurs dans la hauteur des vagues se sont produits, ce qui est essentiel pour la compréhension et la prévision des phénomènes océaniques.

En conclusion, l'algorithme PELT est une méthode prometteuse pour détecter les changements importants dans les données. Il est précis et efficace, ce qui signifie qu'il peut trouver les moments où quelque chose d'important se produit. Son application à des données de la hauteur des vagues a démontré sa capacité à détecter les points de rupture importants et à fournir des informations utiles pour la caractérisation des phénomènes océaniques. Les résultats obtenus suggèrent que l'algorithme PELT pourrait également être utilisé dans d'autres domaines, tels que la finance, la santé ou l'environnement. Il pourrait aider à identifier les moments où des changements significatifs se produisent, ce qui serait précieux pour la prise de décisions ou la détection précoce de problèmes.

Il reste encore des opportunités pour améliorer et étendre l'algorithme PELT. Des recherches futures pourraient explorer des moyens d'optimiser son fonctionnement ou de le combiner avec d'autres méthodes de détection de changements. Cela permettrait d'élargir encore davantage son utilité dans différents domaines et d'apporter de nouvelles perspectives.

A. Annexes

A.1. Annexe 1

```
Created Using changepoint version 2.2.4
Changepoint type      : Change in mean
Method of analysis    : PELT
Test Statistic  : Normal
Type of penalty       : MBIC with value, 25.55158
Minimum Segment Length : 1
Maximum no. of cpts   : Inf
Number of changepoints: 40
```

A.2. Annexe 2

```
Formal class 'cpt' [package "changepoint"] with 12 slots
  ..@ data.set : Time-Series [1:5000] from 1 to 5000: 1.2 1.2 1.2 1.1 1.2 1.1 1.2 1.2 1.2 1.2
  ..@ cpttype  : chr "mean"
  ..@ method   : chr "PELT"
  ..@ test.stat: chr "Normal"
  ..@ pen.type : chr "MBIC"
  ..@ pen.value: num 25.6
  ..@ minseglen: num 1
  ..@ cpts     : int [1:41] 381 413 539 576 626 653 690 775 905 979 ...
  ..@ ncpts.max: num Inf
  ..@ param.est:List of 1
  .. ..$ mean: num [1:41] 2.05 4.8 2.68 6.14 2.9 ...
  ..@ date     : chr "Tue Mar 14 03:56:58 2023"
  ..@ version  : chr "2.2.4"
```

B. Codes

```r
# Exemple sur 3 échantillons (figure 1)

va1 <- rnorm(100, mean = 4, sd = 1)
va2 <- rnorm(100, mean = 8, sd = 1)
va3 <- rnorm(100, mean = 2, sd = 1)

va <- c(va1, va2, va3)
exemple1 <- cpt.mean(va, method = "PELT")
plot(exemple1, col = "blue", lwd = 1, xlab = "x", ylab = "y", main = "Détection de rupture sur
```

```r
# Exemple algorithme PELT avec un K qui ne vérifie pas la condition
library(ggplot2)

visualize_interval_partition <- function(t, T, s, K) {
    # Calculer les coûts partiels et totaux avec la norme 2
    cost_partial1 <- sqrt(sum((1:s)^2))
    cost_partial2 <- sqrt(sum(((s + 1):T)^2))
    cost_total <- sqrt(sum((1:T)^2))

    # Formater les coûts avec deux chiffres après la virgule
    total <- cost_partial1 + cost_partial2 + K
    total <- sprintf("%.2f", total)
    cost_partial1 <- sprintf("%.2f", cost_partial1)
    cost_partial2 <- sprintf("%.2f", cost_partial2)
    cost_total <- sprintf("%.2f", cost_total)

    # Vérifier si l'équation est satisfaite
    condition_satisfied <- (as.numeric(cost_partial1) + as.numeric(cost_partial2) +
        K) <= as.numeric(cost_total)

    # Préparation des données pour la visualisation
    df <- data.frame(x = c(t, s, T), y = rep(0, 3))
    labels <- c("t", "s", "T")

    # Créer le graphique
    ggplot(df, aes(x = x, y = y)) + ggtitle("Exemple où la constante K ne vérifie pas l'équati
        geom_point(size = 4, color = "blue") + geom_text(aes(label = labels), hjust = 0,
        vjust = -1.5, color = "blue") + geom_segment(aes(x = t, xend = T, yend = 0),
        color = "gray", size = 1) + annotate("text", x = (t + s)/2, y = -0.5, label = paste0("
        cost_partial1), color = "black") + annotate("text", x = (s + T)/2, y = -0.5,
        label = paste0("Coût 2 =", cost_partial2), color = "black") + annotate("text",
        x = (t + T)/2, y = -2, label = paste("K =", K), color = "red") + annotate("text",
        x = (t + T)/2, y = -3, label = paste("Condition : Coût 1 + Coût 2 + K <= Coût total"),
        color = "black") + annotate("text", x = (t + T)/2, y = -4, label = paste("Condition :"
        total, ">", cost_total), color = "black") + annotate("text", x = (t + T)/2,
        y = -5, label = ifelse(condition_satisfied, "Condition satisfaite", "Condition non sat
        color = ifelse(condition_satisfied, "darkgreen", "red")) + theme_minimal() +
        xlab("Intervalle") + ylab("") + xlim(t - 1, T + 1) + ylim(-6, 1)
```

```
}

# Exemple d'utilisation
t <- 5
T <- 50
s <- 20
K <- 10

visualize_interval_partition(t, T, s, K)
```

```
# Exemple algorithme PELT avec un K qui vérifie la condition

library(ggplot2)

visualize_interval_partition <- function(t, T, s, K) {
    # Calculer les coûts partiels et totaux avec la fonction de coût spécifiée
    cost_partial1 <- 2 * s
    cost_partial2 <- 3 * (T - s)
    cost_total <- 4 * T
    total <- cost_partial1 + cost_partial2 + K

    # Vérifier si l'équation est satisfaite
    condition_satisfied <- (cost_partial1 + cost_partial2 + K) <= cost_total

    # Formater les coûts avec deux chiffres après la virgule
    total <- sprintf("%.2f", total)
    cost_partial1 <- sprintf("%.2f", cost_partial1)
    cost_partial2 <- sprintf("%.2f", cost_partial2)
    cost_total <- sprintf("%.2f", cost_total)

    # Préparation des données pour la visualisation
    df <- data.frame(x = c(t, s, T), y = rep(0, 3))
    labels <- c("t", "s", "T")

    # Créer le graphique
    ggplot(df, aes(x = x, y = y)) + ggtitle("Exemple où la constante K vérifie l'équation") +
        geom_point(size = 4, color = "blue") + geom_text(aes(label = labels), hjust = 0,
        vjust = -1.5, color = "blue") + geom_segment(aes(x = t, xend = T, yend = 0),
        color = "gray", size = 1) + annotate("text", x = (t + s)/2, y = -0.5, label = paste0("
        cost_partial1), color = "black") + annotate("text", x = (s + T)/2, y = -0.5,
        label = paste0("Coût 2 =", cost_partial2), color = "black") + annotate("text",
        x = (t + T)/2, y = -2, label = paste("K =", K), color = "red") + annotate("text",
        x = (t + T)/2, y = -3, label = paste("Condition : Coût 1 + Coût 2 + K <= Coût total"),
        color = "black") + annotate("text", x = (t + T)/2, y = -4, label = paste("Condition :"
        total, "<=", cost_total), color = "black") + annotate("text", x = (t + T)/2,
        y = -5, label = ifelse(condition_satisfied, "Condition satisfaite", "Condition non sat
        color = ifelse(condition_satisfied, "darkgreen", "red")) + theme_minimal() +
        xlab("Intervalle") + ylab("") + xlim(t - 1, T + 1) + ylim(-6, 1)
}

# Exemple d'utilisation
```

```r
t <- 5
T <- 50
s <- 20
K <- 10

visualize_interval_partition(t, T, s, K)
```

```r
# Visualisation des données sur les vagues

waves <- wave.c44137[1:5000]
plot(waves, type = "l", lwd = 2, col = "blue", axes = FALSE, xlab = "Temps en heure (2005)",
    ylab = "Hauteur en mètres", main = "Hauteur des vagues")
new_labels <- c("janvier", "février", "mars", "avril", "mai", "juin", "juillet",
    "aout")
new_ticks <- c(0, 720, 720 * 2, 3 * 720, 4 * 720, 5 * 720, 6 * 720, 7 * 720)
axis(1, at = new_ticks, labels = new_labels)
axis(2)
```

```r
# Application de l'algorithme sur la hauteur des vagues

resultats <- cpt.mean(waves, method = "PELT")
plot(resultats, type = "l", lwd = 2, col = "gray", axes = FALSE, xlab = "Temps en heure (2005)",
    ylab = "Hauteur en mètres", main = "Détection de rupture sur la hauteur des vagues")
axis(1, at = new_ticks, labels = new_labels)
axis(2)
```

Bibliographie

[AL89] Ivan E. Auger et Charles E. Lawrence. “Algorithms for the optimal identification of segment neighborhoods”. In : *Bulletin of Mathematical Biology* 51.1 (1er jan. 1989), p. 39-54. issn : 1522-9602. doi : 10.1007/BF02458835. url : https://doi.org/10.1007/BF02458835 (visité le 10/05/2023).

[KFE12] R. Killick, P. Fearnhead et I. A. Eckley. “Optimal detection of changepoints with a linear computational cost”. In : *Journal of the American Statistical Association* 107.500 (déc. 2012), p. 1590-1598. issn : 0162-1459, 1537-274X. doi : 10.1080/01621459.2012.737745. arXiv : 1101.1438[q-bio,stat]. url : http://arxiv.org/abs/1101.1438 (visité le 10/05/2023).

[TOV20] Charles Truong, Laurent Oudre et Nicolas Vayatis. “Selective review of offline change point detection methods”. In : *Signal Processing* 167 (fév. 2020), p. 107299. issn : 01651684. doi : 10.1016/j.sigpro.2019.107299. arXiv : 1801.00718[cs,stat]. url : http://arxiv.org/abs/1801.00718 (visité le 01/05/2023).

Webographie

[Gou19] Pêches et Océans Canada Gouvernement du Canada. *Données sur les vagues disponibles en ligne*. Last Modified: 2019-09-26. 26 sept. 2019. url : https://www.meds-sdmm.dfo-mpo.gc.ca/isdm-gdsi/waves-vagues/data-donnees/index-fra.asp (visité le 01/05/2023).

[OT] Florence Offenstein et Julien Thuilliez. *Les vagues Culture Maritime*. url : http://www.culture-maritime.com/fr/text-mme4-cours2 (visité le 01/05/2023).

[Wik23] Wikipédia. *Vague — Wikipédia, l’encyclopédie libre*. Page Version ID: 200831696. 27 jan. 2023. url : https://fr.wikipedia.org/w/index.php?title=Vague&oldid=200831696 (visité le 01/05/2023).