Projet d'Algorithmique

Ton Nom

03 December 2025

Contents

1 Introduction **2**

1.1 Contexte 2

2 Dynamic Time Warping **2**

2.1 Algorithme du DTW 3

2.2 Exemple d'application 4

1 Introduction

Le mathématicien Al-Khwarizmi a rédigé de nombreux travaux qui furent traduits au XIIe siècle. C'est d'ailleurs de son nom que provient le mot « algorithme ». Un algorithme peut être défini comme un enchaînement d'instructions logiques permettant de résoudre un problème. Il constitue la traduction d'un raisonnement en un pseudo-code. Mais dans quelles circonstances construit-on un algorithme, et comment intervient l'algorithmique dans ce processus ? Lorsqu'une problématique nécessite une méthode de calcul efficace, il devient nécessaire d'adopter une démarche algorithmique. L'algorithmique regroupe l'ensemble des méthodes génériques permettant de résoudre des problèmes. Elle s'intéresse aussi à l'étude de la complexité de ces méthodes, c'est-à-dire à leur rapidité, leur efficacité et leur capacité à s'adapter à différents cas. Autrement dit, elle ne se limite pas à la simple écriture d'un algorithme, mais vise à concevoir la meilleure approche possible pour atteindre un résultat fiable et optimisé. Aujourd'hui, au XXIe siècle, de nombreux mathématiciens et informaticiens ont inventé, amélioré et publié diverses méthodes de résolution. Ces avancées nous permettent de disposer d'outils conceptuels pour résoudre des problèmes complexes.

La résolution d'une problématique passe d'abord par l'évaluation de la complexité du problème, puis par le choix d'une méthode adaptée, d'où l'utilisation de l'algorithmique. Vient ensuite la construction du programme à l'aide de l'algorithme et en utilisant un langage de programmation choisi. Après l'exécution du programme, un résultat concret est obtenu, traduisant la pertinence de la méthode employée.

1.1 Contexte

Une série temporelle est une suite chronologique de valeurs numériques. Les séries temporelles sont utilisées dans plusieurs domaines, comme en médecine, par exemple pour le suivi d'une constante vitale au cours du temps, ou encore en finance pour le suivi du cours d'une action. Dans ces deux disciplines, l'analyse de ces séries temporelles peut être cruciale, notamment pour le diagnostic d'une anomalie ou de la détection d'un comportement particulier. Dans notre cas, nous allons nous intéresser à une méthode permettant de classifier les séries temporelles. Dans un premier temps, nous allons utiliser deux algorithmes qui vont nous permettre de comparer deux séries entre elles. Plus précisément, nous allons utiliser l'algorithme DTW, puis une amélioration de cet algorithme appelée FastDTW. Enfin, nous choisirons le meilleur algorithme pour la classification.

2 Dynamic Time Warping

L'algorithme Dynamic Time Warping (DTW) ou Déformation Temporelle Dynamique, permet de mesurer la similarité entre deux séries temporelles qui peuvent différer en vitesse ou en phase.

Cette méthode aligne de manière non linéaire deux séquences temporelles pour obtenir la correspondance optimale entre leurs éléments.

Soient deux séries temporelles :

$$X = (x_1, x_2, ..., x_n) \quad \text{et} \quad Y = (y_1, y_2, ..., y_m)$$

On définit la distance D entre deux points correspondants :

$$D(i, j) = (x_i - y_j)^2$$

Le but du DTW est de construire une matrice de coût cumulatif C, où chaque élément $C(i, j)$ représente le coût minimal d'alignement des sous-séquences $(x_1, ..., x_i)$ et $(y_1, ..., y_j)$.
Cette matrice est calculée récursivement selon :

$$C(i,j) = \begin{cases} D(1,1), & \text{si } i = 1, j = 1, \\ D(i,1) + C(i-1,1), & \text{si } j = 1, i > 1, \\ D(1,j) + C(1,j-1), & \text{si } i = 1, j > 1, \\ D(i,j) + \min\left(C(i-1,j),\ C(i,j-1),\ C(i-1,j-1)\right), & \text{sinon.} \end{cases}$$

L'objectif de la méthode DTW est de trouver le chemin optimal qui minimise la distance totale entre les deux séries temporelles.

La distance DTW finale est alors donnée par :

$$DTW(n, m) = \min_{\pi} \sum_{(i,j)\in\pi} D(i, j)$$

où π désigne l'ensemble des chemins de warping valides reliant les points $(1, 1)$ et (n, m).

Dans la pratique, cette distance correspond à la valeur du dernier élément de la matrice de coût cumulatif :

$$DTW(X, Y) = C(n, m)$$

2.1 Algorithme du DTW

L'algorithme DTW repose sur la programmation dynamique.

Il consiste à remplir progressivement la matrice de coût cumulé C.

```
Algorithme DTW(x, y):
1. Initialiser D[i,j] = |x_i - y_j|
2. Initialiser C[1,1] = D[1,1]
3. Remplir la première ligne et la première colonne de C
4. Pour i=2..n, j=2..m :
     C[i,j] = D[i,j] + min(C[i-1,j-1], C[i-1,j], C[i,j-1])
5. Rétrotraçage pour obtenir le chemin optimal
6. Retourner C, traceback, path
```

Path correspond au chemin optimal. La complexité en temps de l'algorithme DTW est de l'ordre de $O(NM)$.

2.2 Exemple d'application

Considérons les deux séries temporelles suivantes :

$$X = (1, 2, 3, 4) \quad \text{et} \quad Y = (1, 1, 2, 3, 5)$$

Calcul de la matrice de distances locales $D(i, j)$:

$$D = \begin{bmatrix} 0 & 0 & 1 & 4 & 16 \\ 1 & 1 & 0 & 1 & 9 \\ 4 & 4 & 1 & 0 & 4 \\ 9 & 9 & 4 & 1 & 1 \end{bmatrix}$$

Calcul de la matrice de coût cumulatif $C(i, j)$:

En remplissant la matrice pas à pas, on obtient :

$$C = \begin{bmatrix} 0 & 0 & 1 & 5 & 21 \\ 1 & 1 & 0 & 1 & 10 \\ 5 & 5 & 1 & 0 & 4 \\ 14 & 14 & 5 & 1 & 2 \end{bmatrix}$$

La distance DTW entre X et Y est donnée par le dernier élément de la matrice :

$$DTW(X, Y) = C(4, 5) = 2$$

Cette valeur faible indique que les deux séries sont similaires, même si elles ne sont pas parfaitement alignées temporellement.

Le DTW a « déformé » le temps pour trouver un alignement optimal entre les points de X et Y.

```r
library(ProjetM2Algo)
x <- c(1, 3, 4, 9)
y <- c(1, 3, 7, 8, 9)
dtw_result <- dtw(x, y)
dtw_result
```

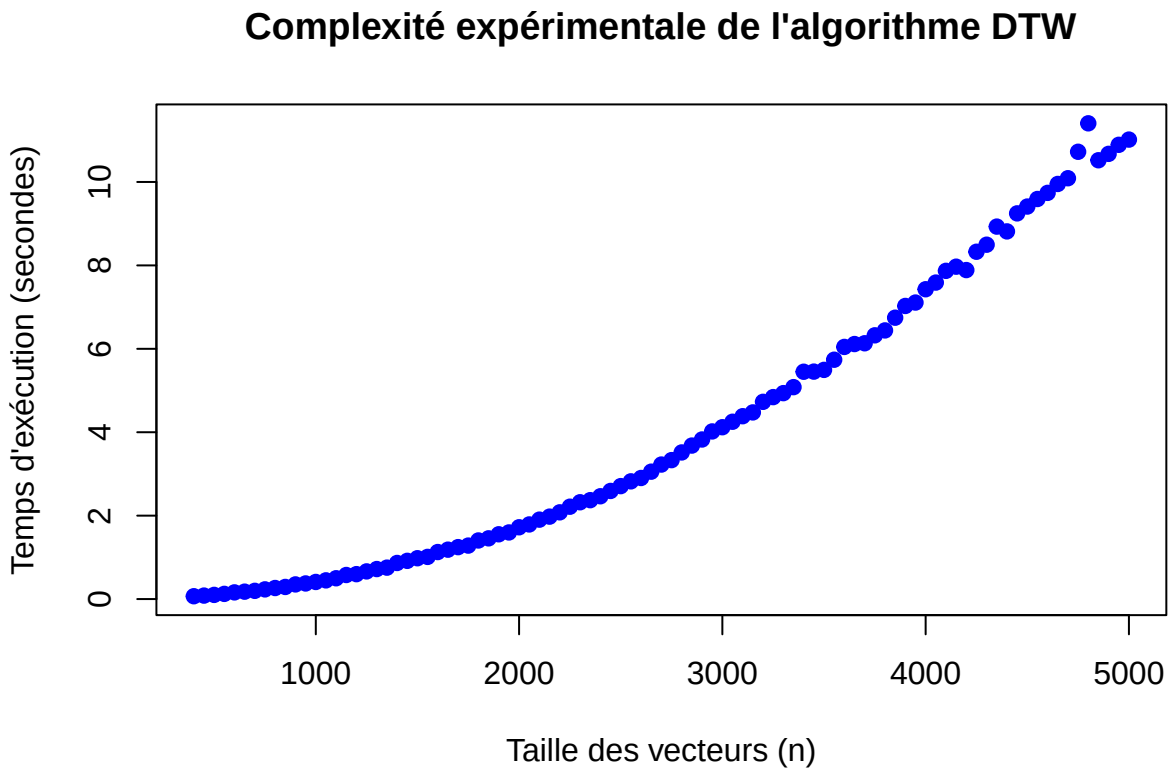
```
## $DTW_matrix
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    2    8   15   23
## [2,]    2    0    4    9   15
## [3,]    5    1    3    7   12
## [4,]   13    7    3    4    4
##
## $traceback_matrix
```

```
##      [,1] [,2]   [,3]   [,4]   [,5]
## [1,] ""   "left" "left" "left" "left"
## [2,] "up" "diag" "left" "left" "left"
## [3,] "up" "up"   "diag" "left" "left"
## [4,] "up" "up"   "diag" "diag" "left"
##
## $path
## $path[[1]]
## [1] 1 1
##
## $path[[2]]
## [1] 2 2
##
## $path[[3]]
## [1] 3 3
##
## $path[[4]]
## [1] 4 4
##
## $path[[5]]
## [1] 4 5
```

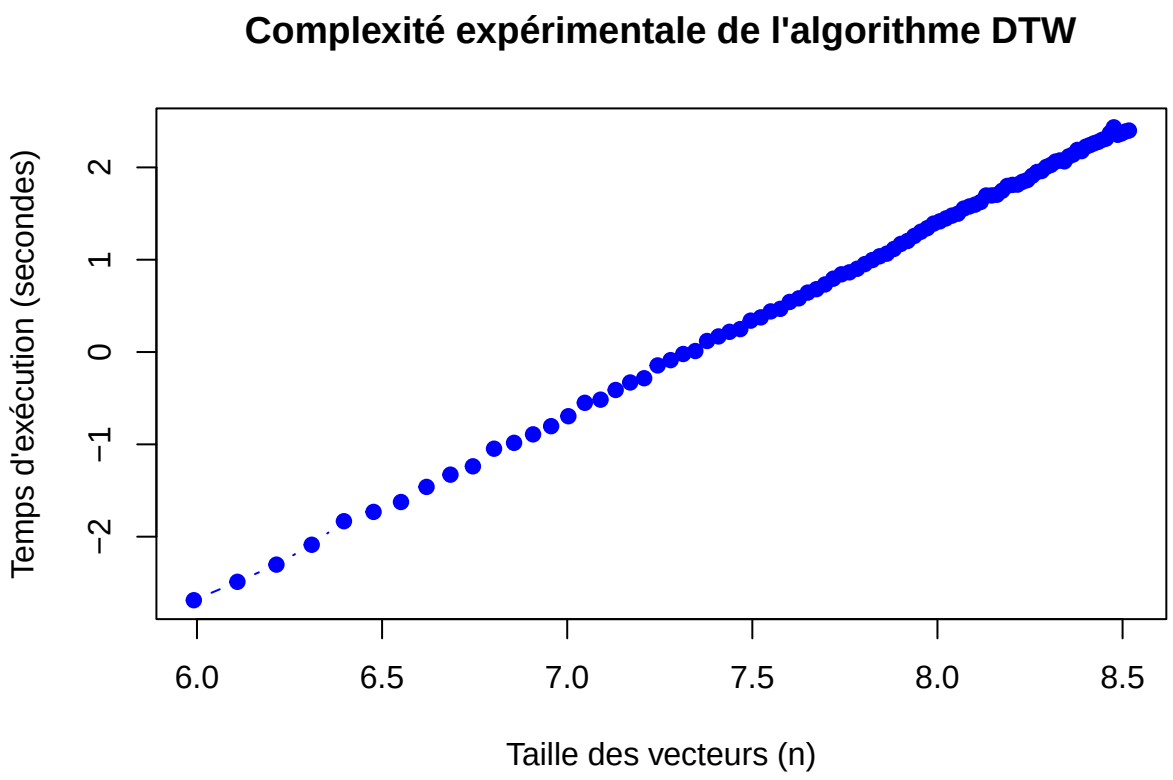
Maintenant nous allons essayer de retrouver la complexité de l'algorithme.

```r
# Vecteurs de tailles à tester
taille_vecteurs <- seq(400, 5000, by = 50)

# vecteur vide temps pour garder le temps d'exécution
temps <- numeric(length(taille_vecteurs))

# Boucle de test
for (k in seq_along(taille_vecteurs)) {
  n <- taille_vecteurs[k]
  x <- runif(n)  # crée un vecteur aléatoire
  y <- runif(n)

  temps[k] <- system.time(dtw(x, y))["elapsed"]
}

# Regrouper les résultats
resultats <- data.frame(
  Taille = taille_vecteurs,
  Temps = temps
)
```

```r
plot(resultats$Taille, resultats$Temps, type = "b", pch = 19, col = "blue",
     main = "Complexité expérimentale de l'algorithme DTW",
     xlab = "Taille des vecteurs (n)",
     ylab = "Temps d'exécution (secondes)")
```

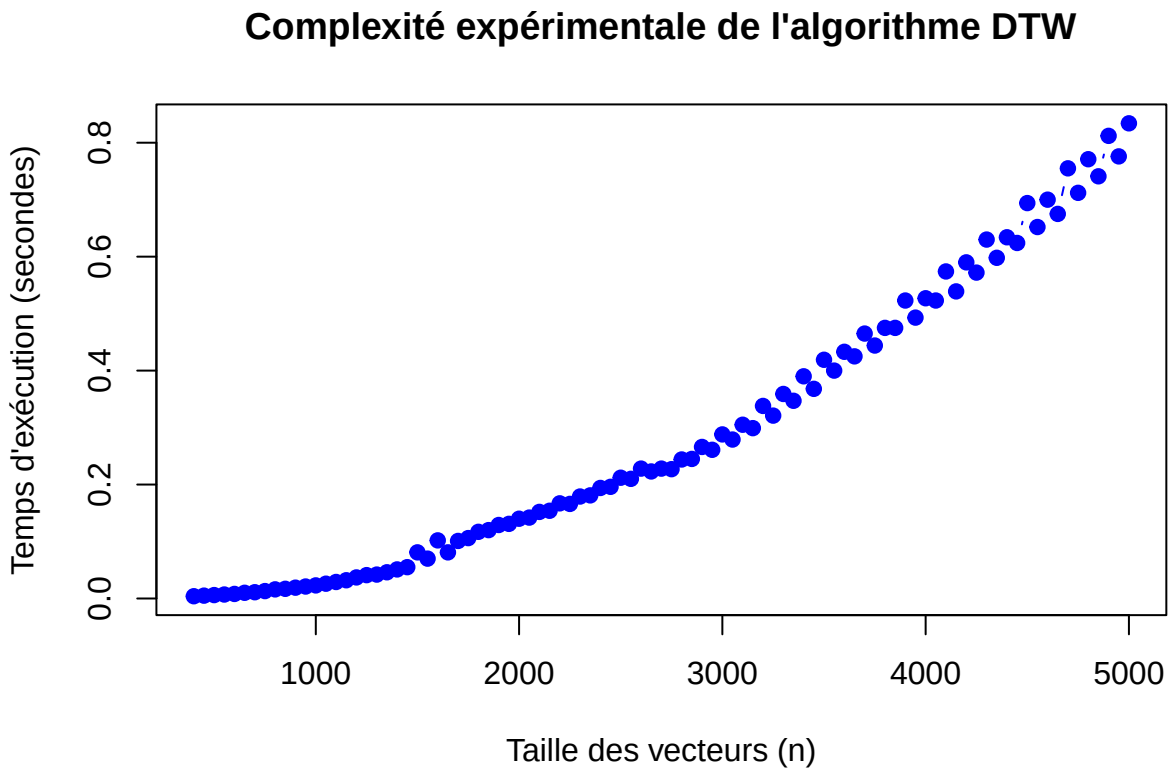



```r
plot(log(resultats$Taille), log(resultats$Temps), type = "b", pch = 19, col = "blue",
     main = "Complexité expérimentale de l'algorithme DTW",
     xlab = "Taille des vecteurs (n)",
     ylab = "Temps d'exécution (secondes)")
```

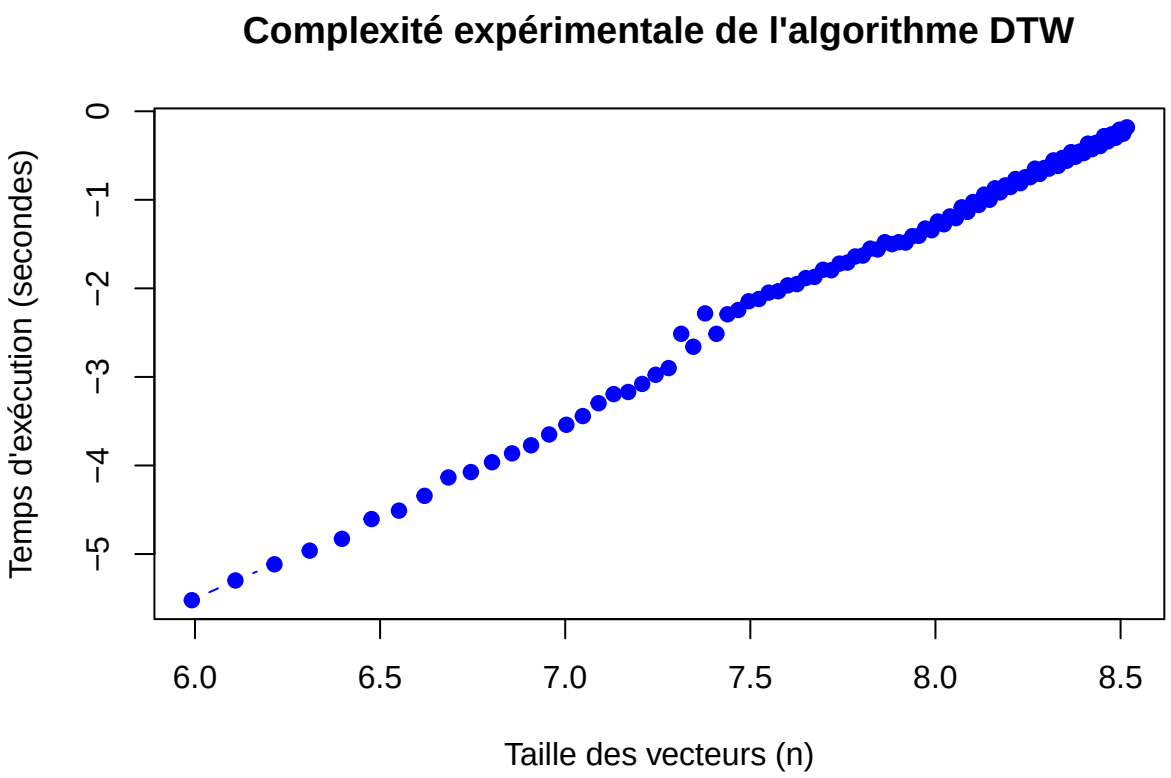



```r
# Vecteurs de tailles à tester
taille_vecteurs <- seq(400, 5000, by = 50)

# vecteur vide temps pour garder le temps d'exécution
temps <- numeric(length(taille_vecteurs))

# Boucle de test
for (k in seq_along(taille_vecteurs)) {
  n <- taille_vecteurs[k]
  x <- runif(n)  # crée un vecteur aléatoire
  y <- runif(n)

  temps[k] <- system.time(dtw_rcpp(x, y))["elapsed"]
}

# Regrouper les résultats
resultats <- data.frame(
  Taille = taille_vecteurs,
  Temps = temps
)
```

```r
plot(resultats$Taille, resultats$Temps, type = "b", pch = 19, col = "blue",
     main = "Complexité expérimentale de l'algorithme DTW",
     xlab = "Taille des vecteurs (n)",
     ylab = "Temps d'exécution (secondes)")
```




```r
plot(log(resultats$Taille), log(resultats$Temps), type = "b", pch = 19, col = "blue",
     main = "Complexité expérimentale de l'algorithme DTW",
     xlab = "Taille des vecteurs (n)",
     ylab = "Temps d'exécution (secondes)")
```


Complexité expérimentale de l'algorithme DTW
Temps d'exécution (secondes)
0
−1
−2
−3
−4
−5
6.0
6.5
7.0
7.5
8.0
8.5
Taille des vecteurs (n)