Requirement

To be able to launch this notebook, you will need a version of python 3.6.9 with these mandatory libraries:

pip install tensorflow-gpu==2.4.1 keras==2.2.4 numpy==1.19.5 matplotlib==2.2.2 scikit-image==0.15.0 tqdm

Then, to select this python for the project, you will have to go to the tab:

Tools →Project Options..→Python

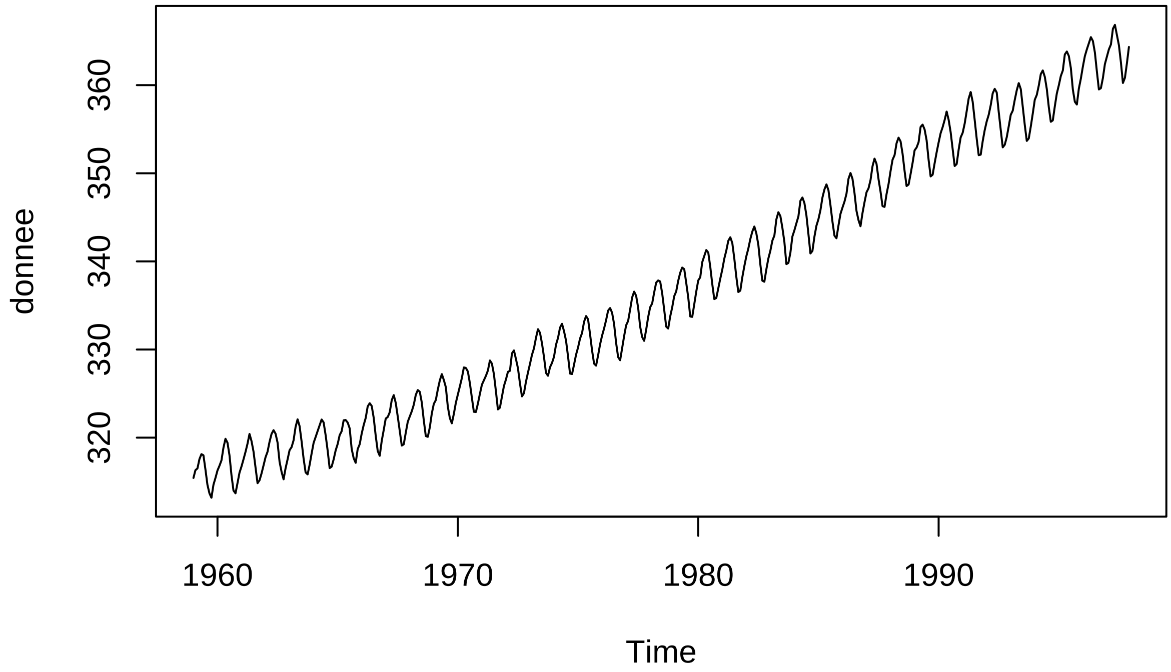
The Reticulate library was compiled under version 4.3.1 of R

Library

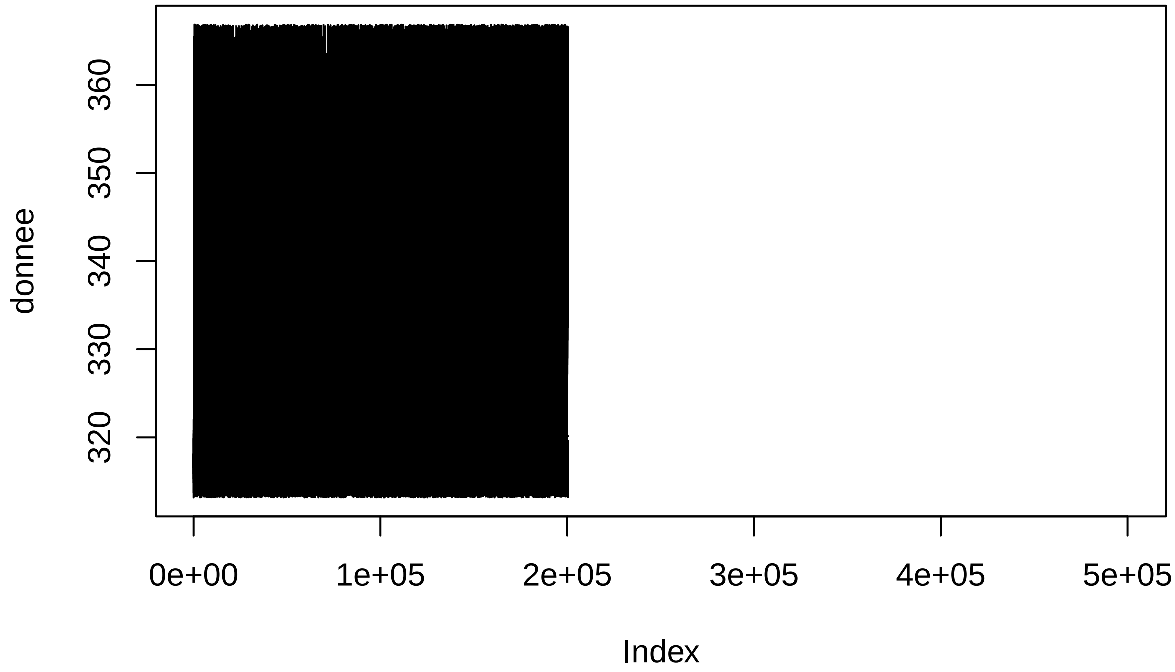
```
library(TSA)
library(DTWBI)
library(dtw)
library(ggplot2)
library(gridExtra)
library(cluster)
library(factoextra)
library(dtwclust)
library(htmlwidgets)
library(plotly)
library(tidyr)
library(reshape2)
library(tidyverse)
library(reticulate)
library(knitr)

rm(list = ls())

source("R_function/EC_completion.r")
source("R_function/EC_compareCourbe.r")
source("R_function/globalF.r")
source("R_function/compute.erreurRMSE.r")
source_python("Python_function/TimeWarp.py")
source("R_function/compute.DistanceMatrix.r")
```

TimeWarp

```
donnee <- datasets::co2
plot(donnee,type = "l")
```



```
donnee <- unlist(TimeWarp(
  data = r_to_py(donnee),
  nbpts = 500000,
  seq_len = 467,
  condition = 8,
  verbose = FALSE
))
plot(donnee, type = "l")
```



Finding Feasible Windows and keep inputation result

The method for selecting a window has changed again. 1. We just keep all the windows that meet the cosine criterion, which is 0.95 2. We are going to use the "compute.indicatorComp()" function to refine the windows that we are going to keep, this will allow us to have curves that are more like the reference one. There are 8 conditions in the function and in the code, we decide to keep the windows meeting 7 or more criteria.

```
gapTaille <- 7
  gapStart <- 400
  dataModif <-
    gapCreation(donnee, gapTaille / length(donnee), gapStart)$output_vector
  queryTaille <- 12
  featureRef <-
    globalfeatures(dataModif[(gapStart - queryTaille):(gapStart - 1)])
  queryRef <- dataModif[(gapStart - queryTaille):(gapStart - 1)]
  fenetresViable <- data.frame("queryRef" = queryRef)
  repRef <- donnee[gapStart:(gapStart + gapTaille - 1)]
  reponseViable <- data.frame("repRef" = repRef)
  debut <- 1
  fin <- debut + queryTaille

  # if (length(data) < 10000) {
  #   step_threshold <- 1
```

```r
# } else{
#   if (length(data) > 1000000) {
#     step_threshold <- 10
#   } else{
#     step_threshold <- 5
#   }
# }

threshold_cos <- 0.95

#' Param to lower if there is a problem during clustering.
#' The lower the value, the longer it will take
step_threshold <- 5

cos_score <- c()
deb_vect <- c()

while (((fin + queryTaille + gapTaille) < length(donnee))) {
  if (!(
    debut %in% seq(
      gapStart - queryTaille - queryTaille,
      gapStart + gapTaille + queryTaille
    )
  )) {
    featureTemp <- globalfeatures(dataModif[debut:(fin - 1)])
    queryTemp <- dataModif[debut:(fin - 1)]
    cosCompare <- abs(cosine(featureRef, featureTemp))

    if (!is.na(cosCompare) && cosCompare >= threshold_cos) {
      cos_score <- c(cos_score, cosCompare)
      deb_vect <- c(deb_vect, debut)
    }
  }
  debut <- debut + step_threshold
  fin <- debut + queryTaille
}

for (i in 1:length(deb_vect)) {
  debut <- deb_vect[i]
  fin <- debut + queryTaille
  if (compute.indicateurComp(queryRef, dataModif[debut:(fin - 1)])[12] >= 7) {
    fenetresViable <- cbind(fenetresViable,
                            queryTemp <- dataModif[debut:(fin - 1)])
    colnames(fenetresViable)[ncol(fenetresViable)] <-
      paste0("Debut = ", deb_vect[i])

    repTemp <- dataModif[fin:(fin + gapTaille - 1)]
    reponseViable <- cbind(reponseViable, repTemp)
    colnames(reponseViable)[ncol(reponseViable)] <-
      paste0("Debut = ", deb_vect[i])
  }
}
fenetresViable <- subset(fenetresViable, select = -1)
```

```r
reponseViable <- subset(reponseViable, select = -1)
```

```
##    Debut = 11611 Debut = 19071 Debut = 24196 Debut = 32141 Debut = 35406
## 1       358.6033      358.6857      358.6418      358.3141      358.2890
## 2       359.1032      358.8941      359.0024      359.1331      359.1324
## 3       358.0184      357.4707      357.7575      358.2535      358.2379
## 4       356.0684      355.3439      355.7326      356.3193      356.2616
## 5       354.0347      353.2570      353.6792      354.2006      354.1138
## 6       352.2360      352.0726      352.0563      352.3063      352.2096
## 7       352.1016      352.7157      352.2273      352.1014      352.1051
## 8       353.3657      354.1616      353.7008      353.4089      353.5163
## 9       354.6236      355.3248      354.9090      354.6910      354.7878
## 10      355.6397      356.2262      355.8685      355.7136      355.7961
## 11      356.4291      357.1580      356.5971      356.4968      356.5627
## 12      357.4055      358.4054      357.6353      357.5150      357.6143
##    Debut = 48976 Debut = 58741 Debut = 86791 Debut = 100331 Debut = 111986
## 1       358.5105      358.6116      358.1654       358.1020       358.6395
## 2       359.1892      359.0817      359.1552       359.0552       358.9224
## 3       358.0811      357.9749      358.0345       358.3227       357.4590
## 4       356.0305      356.0349      355.6951       356.3590       355.2761
## 5       353.8957      354.0282      353.3602       354.1458       353.1594
## 6       352.0510      352.2676      352.0757       352.1722       352.0766
## 7       352.1479      352.0989      352.9615       352.1086       352.8335
## 8       353.6871      353.2450      354.4960       353.6611       354.2630
## 9       354.9450      354.4721      355.6948       354.9483       355.4037
## 10      355.9389      355.4702      356.5855       355.9562       356.2808
## 11      356.7465      356.2572      357.8122       356.7878       357.2238
## 12      357.9100      357.0838      359.1698       357.9747       358.4626
##    Debut = 139076 Debut = 159626 Debut = 202126 Debut = 203066 Debut = 217081
## 1        357.7420       358.0503       358.5523       357.8245       358.5825
## 2        358.9729       359.0694       359.2133       359.0227       359.0177
## 3        358.3021       358.2457       358.2468       358.2136       357.6460
## 4        356.0669       356.1179       356.5153       355.8619       355.4674
## 5        353.6370       353.8184       354.6184       353.4224       353.3135
## 6        352.0680       352.0580       352.8564       352.0762       352.0724
## 7        352.7841       352.4289       352.0779       353.0300       352.7545
## 8        354.3586       353.9898       352.6615       354.5935       354.2282
## 9        355.5852       355.2340       353.9613       355.8009       355.4054
## 10       356.4969       356.1884       355.0425       356.7096       356.3084
## 11       357.6499       357.1433       355.9187       358.0188       357.3084
## 12       359.0912       358.4308       356.5923       359.2563       358.6303
##    Debut = 241811 Debut = 257221 Debut = 260026 Debut = 263316 Debut = 275441
## 1        358.5917       357.3796       358.5807       357.0696       358.1915
## 2        359.0982       358.8334       359.1876       358.6364       359.1931
## 3        357.9797       358.4466       358.2056       358.7859       357.8731
## 4        356.0112       356.2289       356.4426       356.9304       355.4823
## 5        353.9739       353.6937       354.5501       354.4344       353.1459
## 6        352.1919       352.0688       352.8052       352.1604       352.0832
## 7        352.1023       352.8645       352.0792       352.2958       353.1557
## 8        353.3565       354.4684       352.6840       354.0161       354.6509
## 9        354.5849       355.7031       353.9694       355.3645       355.8083
## 10       355.5775       356.6067       355.0396       356.3725       356.6732
## 11       356.3527       357.8612       355.9085       357.5378       357.9154
## 12       357.2439       359.1869       356.5730       359.0740       359.1941
##    Debut = 287116 Debut = 300666 Debut = 322621 Debut = 329146 Debut = 350161
## 1        358.0653       358.5509       357.7005       357.9542       357.1550
## 2        359.0757       359.1667       358.9124       359.0071       358.7212
## 3        358.2389       358.1171       358.4448       358.3588       358.6077
## 4        356.1121       356.1536       356.4224       356.3699       356.5165
## 5        353.8207       354.1172       354.0441       354.1083       353.9390
## 6        352.0577       352.3093       352.0536       352.1055       352.0617
## 7        352.4113       352.0990       352.3757       352.1473       352.7017
## 8        353.9678       353.2864       353.9991       353.7326       354.3535
## 9        355.2100       354.5436       355.2854       355.0083       355.6282
## 10       356.1647       355.5603       356.2625       356.0005       356.5636
## 11       357.1004       356.3536       357.3047       356.8452       357.8110
## 12       358.3665       357.2688       358.7006       358.0305       359.1790
##    Debut = 387951 Debut = 394546 Debut = 457111 Debut = 479501 Debut = 495876
## 1        358.3629       357.7467       357.6636       358.6020       358.1252
## 2        359.1780       358.9441       358.8955       359.0469       359.1148
## 3        358.1680       358.3838       358.4686       357.8230       358.1652
## 4        356.1443       356.2788       356.4678       355.7922       355.9345
## 5        354.0245       353.8680       354.0952       353.7325       353.6265
## 6        352.1755       352.0598       352.0519       352.0542       352.0653
## 7        352.1043       352.5647       352.3259       352.1605       352.6442
## 8        353.4344       354.1806       353.9498       353.6254       354.1936
## 9        354.6611       355.4525       355.2380       354.8173       355.4197
## 10       355.6407       356.4083       356.2189       355.7685       356.3492
## 11       356.3974       357.5466       357.2283       356.5003       357.4086
## 12       357.2970       359.0403       358.5876       357.4593       358.8018
```

```
##   Debut = 11611 Debut = 19071 Debut = 24196 Debut = 32141 Debut = 35406
## 1      358.6589      359.3411      358.9175      358.8011      358.9266
## 2      359.4191      359.3522      359.5036      359.4724      359.5200
## 3      359.3019      357.8754      359.2472      359.2618      359.2239
## 4      357.6593      355.7163      357.4491      357.4686      357.2676
## 5      355.5624      353.6695      355.4653      355.4274      355.2528
## 6      353.5498      353.1308      353.5604      353.4668      353.3058
## 7      353.1453      353.8187      353.1261      353.1484      353.1701
##   Debut = 48976 Debut = 58741 Debut = 86791 Debut = 100331 Debut = 111986
## 1      359.1620      358.1604      359.4478       359.1890       359.3507
## 2      359.4874      359.1887      358.1678       359.4659       359.3564
## 3      358.5509      359.5245      355.7766       358.4464       357.9758
## 4      356.2616      359.0920      353.5297       356.1902       355.8886
## 5      354.1265      357.0948      353.1802       354.0912       353.9292
## 6      353.0771      355.2784      354.0503       353.0754       353.0779
## 7      353.7045      353.5122      355.4329       353.6735       353.6127
##   Debut = 139076 Debut = 159626 Debut = 202126 Debut = 203066 Debut = 217081
## 1       359.5228       359.3648       357.5548       359.3731       359.4407
## 2       358.6560       359.3237       358.7294       357.7429       359.2603
## 3       356.3054       357.6796       359.4003       355.3834       357.3114
## 4       354.1467       355.5118       359.3554       353.1445       355.1468
## 5       353.0734       353.4522       358.1690       353.2457       353.0536
## 6       353.6785       353.1637       356.2320       354.2570       353.2295
## 7       354.8040       353.9151       354.4196       355.6504       354.1573
##   Debut = 241811 Debut = 257221 Debut = 260026 Debut = 263316 Debut = 275441
## 1       358.3862       359.4395       357.5119       359.5058       359.4427
## 2       359.2848       358.1758       358.6560       358.3823       358.2328
## 3       359.4383       355.8532       359.3642       355.8819       355.9327
## 4       358.5729       353.6932       359.3917       353.5480       353.7908
## 5       356.5622       353.1403       358.4090       353.1896       353.1242
## 6       354.7152       353.8759       356.4911       354.1226       353.8261
## 7       352.9450       355.0834       354.7174       355.5772       355.0130
##   Debut = 287116 Debut = 300666 Debut = 322621 Debut = 329146 Debut = 350161
## 1       359.3361       358.4501       359.4944       359.2017       359.4363
## 2       359.3506       359.3229       359.1962       359.4644       358.1026
## 3       357.8474       359.3952       356.8621       358.4976       355.7330
## 4       355.6840       358.2643       354.6446       356.3002       353.5195
## 5       353.6414       356.2094       353.0090       354.2715       353.1744
## 6       353.1329       354.2915       353.5180       353.0369       354.0039
## 7       353.8144       353.0193       354.6232       353.5190       355.3086
##   Debut = 387951 Debut = 394546 Debut = 457111 Debut = 479501 Debut = 495876
## 1       358.4283       359.5327       359.4438       358.6498       359.5222
## 2       359.2884       358.6046       359.2440       359.3816       359.1824
## 3       359.4479       356.1481       357.1564       359.3639       356.8363
## 4       358.7064       353.8685       354.9551       358.1937       354.6610
## 5       356.7689       353.1345       352.9570       356.2478       353.0005
## 6       355.0162       353.9271       353.3423       354.4410       353.4741
## 7       353.3150       355.2694       354.3367       352.9794       354.5330
```

Calculation of DTW and Soft-DTW distance matrices

```
g = 0.001
  print(length(fenetresViable))
```

```
## [1] 30
```

```
  matriceDTW <-
    compute.DistanceMatrixDTW(fenetresViable, normalize = FALSE)
  matriceSDTW <-
    compute.DistanceMatrixSDTW(fenetresViable, g, normalize = FALSE)
  miniDTW <- min(matriceDTW)

maxiDTW <- max(matriceDTW)
matriceDTW <- matriceDTW - miniDTW
matriceDTW <- matriceDTW / (maxiDTW - miniDTW)

miniSDTW <- min(matriceSDTW)
maxiSDTW <- max(matriceSDTW)
matriceSDTW <- matriceSDTW - miniSDTW
matriceSDTW <- matriceSDTW / (maxiSDTW - miniSDTW)

# print(matriceDTW)
# print(matriceSDTW)
```

Clustering according to PAM

```
nbclusterDTW <-
  fviz_nbclust(t(fenetresViable),
               pam,
               diss = matriceDTW,
               method = "silhouette", )
nbclusterDTW <- nbclusterDTW$data$y
nbclusterDTW <- which.max(nbclusterDTW)
nbclusterSDTW <-
  fviz_nbclust(t(fenetresViable),
               pam,
               diss = matriceSDTW,
               method = "silhouette", )
nbclusterSDTW <- nbclusterSDTW$data$y
nbclusterSDTW <- which.max(nbclusterSDTW)

resultatPamDTW <-
  pam(matriceDTW,
      nbclusterDTW,
      diss = TRUE,
      cluster.only = TRUE)
resultatPamSDTW <-
  pam(matriceSDTW,
      nbclusterSDTW,
      diss = TRUE,
      cluster.only = TRUE)
```

Plot

Criteria

```
#   Partie 1: Cluster ayant la moyenne DTW/SDTW la plus faible
avgClusterDTW <- rep(0, times = nbclusterDTW)
avgClusterSDTW <- rep(0, times = nbclusterSDTW)

for (i in 1:length(fenetresViable)) {
  avgClusterDTW[resultatPamDTW[i]] <-
    avgClusterDTW[resultatPamDTW[i]] + matriceDTW[i]
  avgClusterSDTW[resultatPamSDTW[i]] <-
    avgClusterSDTW[resultatPamSDTW[i]] + matriceSDTW[i]
}

for (i in 1:nbclusterDTW) {
  avgClusterDTW[i] <-
    avgClusterDTW[i] / table(resultatPamDTW)[i]
}

for (i in 1:nbclusterSDTW) {
  avgClusterSDTW[i] <-
```

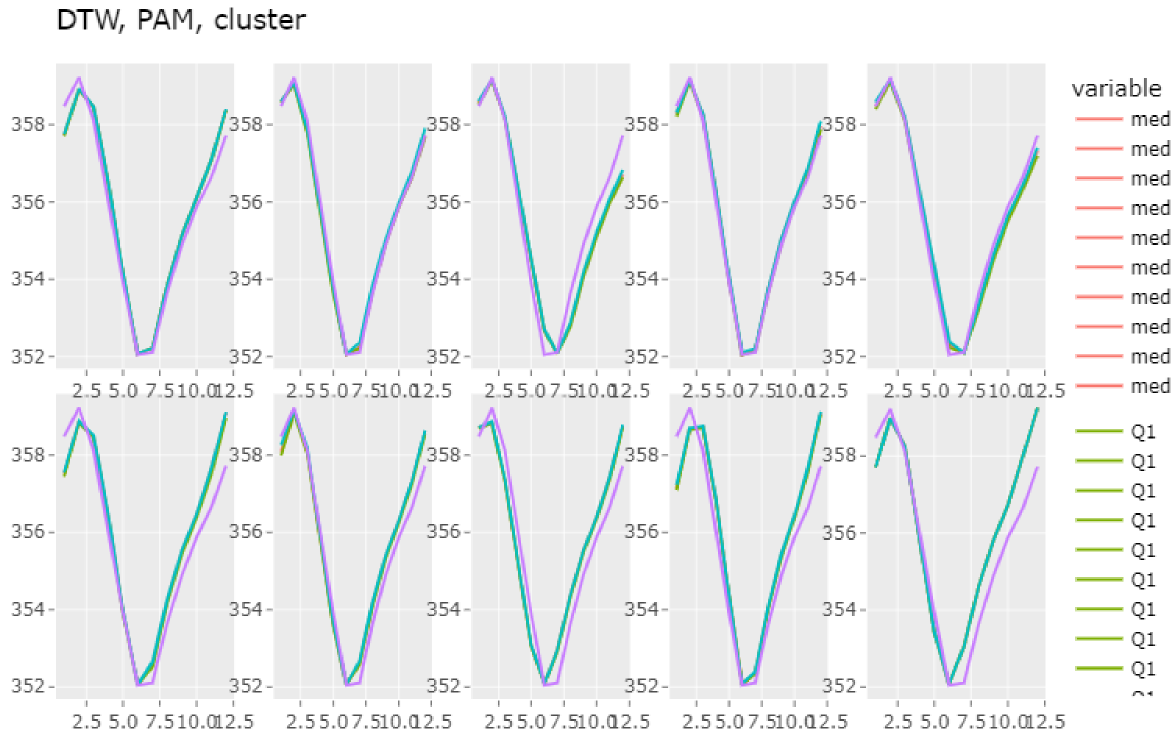


Figure 1: DTW Cluster

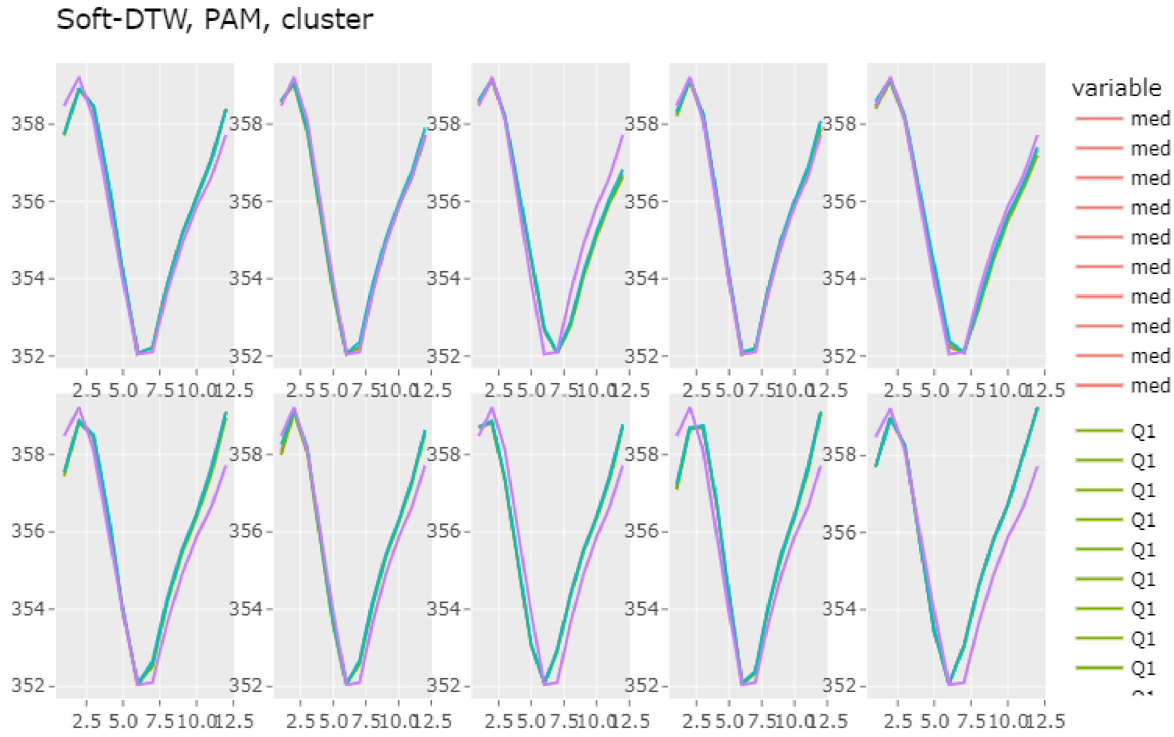


Figure 2: SDTW Cluster

```
    avgClusterSDTW[i] / table(resultatPamSDTW)[i]
}

cat("\nPartie 1\n")
```

```
##
## Partie 1
```

```
print(paste(
  "Pour DTW, le cluster",
  which.min(avgClusterDTW),
  "a la moyenne de coût DTW la plus faible"
))
```

```
## [1] "Pour DTW, le cluster 1 a la moyenne de coût DTW la plus faible"
```

```
print(paste(
  "Pour SDTW, le cluster",
  which.min(avgClusterSDTW),
  "a la moyenne de coût DTW la plus faible"
))
```

```
## [1] "Pour SDTW, le cluster 1 a la moyenne de coût DTW la plus faible"
```

```
#   Partie 2: Cluster le plus représenté
cat("\nPartie 2\n")
```

```
##
## Partie 2
```

```
print(paste("Pour DTW, le cluster",
            which.max(table(resultatPamDTW)),
            "a le plus de points"))
```

```
## [1] "Pour DTW, le cluster 1 a le plus de points"
```

```
print(paste("Pour SDTW, le cluster",
            which.max(table(resultatPamSDTW)),
            "a le plus de points"))
```

```
## [1] "Pour SDTW, le cluster 1 a le plus de points"
```

```
#   Partie 3: Cluster le moyenne d'érreur quadratique la plus basse
avgQuadClusterDTW <- rep(0, times = nbclusterDTW)
avgQuadClusterSDTW <- rep(0, times = nbclusterSDTW)

for (i in 1:length(fenetresViable)) {
  avgQuadClusterDTW[resultatPamDTW[i]] <-
    avgQuadClusterDTW[resultatPamDTW[i]] + rmse(queryRef, fenetresViable[, i])
  avgQuadClusterSDTW[resultatPamSDTW[i]] <-
    avgQuadClusterSDTW[resultatPamSDTW[i]] + rmse(queryRef, fenetresViable[, i])
}

for (i in 1:nbclusterDTW) {
  avgQuadClusterDTW[i] <-
    avgQuadClusterDTW[i] / table(resultatPamDTW)[i]
}

for (i in 1:nbclusterSDTW) {
  avgQuadClusterSDTW[i] <-
    avgQuadClusterSDTW[i] / table(resultatPamSDTW)[i]
}

cat("\nPartie 3\n")
```

```
##
## Partie 3
```

```
print(paste(
  "Pour DTW, le cluster",
  which.min(avgQuadClusterDTW),
  "a la moyenne RMSE la plus faible"
))
```

```
## [1] "Pour DTW, le cluster 4 a la moyenne RMSE la plus faible"
```

```
print(paste(
  "Pour SDTW, le cluster",
  which.min(avgQuadClusterSDTW),
  "a la moyenne RMSE la plus faible"
))
```

```
## [1] "Pour SDTW, le cluster 4 a la moyenne RMSE la plus faible"
```

```
# Partie 4:
avgAmpAvgClusterDTW <- rep(0, times = nbclusterDTW)
avgAmpAvgClusterSDTW <- rep(0, times = nbclusterSDTW)

for (i in 1:length(reponseViable)) {
  if (queryRef[1] >= fenetresViable[1, i]) {
    avgAmpAvgClusterDTW[resultatPamDTW[i]] <-
      avgAmpAvgClusterDTW[resultatPamDTW[i]] + dtw_basic(queryRef,
                                                         fenetresViable[, i] + mean(abs(queryRef - fene
    avgAmpAvgClusterSDTW[resultatPamSDTW[i]] <-
      avgAmpAvgClusterSDTW[resultatPamSDTW[i]] + sdtw(queryRef,
                                                      fenetresViable[, i] + abs(queryRef - fenetresViab
                                                      gamma = g)
  } else{
    avgAmpAvgClusterDTW[resultatPamDTW[i]] <-
      avgAmpAvgClusterDTW[resultatPamDTW[i]] + dtw_basic(queryRef,
                                                         fenetresViable[, i] - abs(queryRef - fenetresVi
    avgAmpAvgClusterSDTW[resultatPamSDTW[i]] <-
      avgAmpAvgClusterSDTW[resultatPamSDTW[i]] + sdtw(queryRef,
                                                      fenetresViable[, i] - abs(queryRef - fenetresViabl
                                                      gamma = g)
  }
}

for (i in 1:nbclusterDTW) {
  avgAmpAvgClusterDTW[i] <-
    avgAmpAvgClusterDTW[i] / table(resultatPamDTW)[i]
}

for (i in 1:nbclusterSDTW) {
  avgAmpAvgClusterSDTW[i] <-
    avgAmpAvgClusterSDTW[i] / table(resultatPamSDTW)[i]
}

cat("\nPartie 4\n")
```

```
##
## Partie 4
```

```
print(
  paste(
    "Pour DTW, le cluster",
    which.min(avgAmpAvgClusterDTW),
    "a la moyenne de coût DTW la plus faible quand on met le tout les points de la query temp à une dis
  )
)
```

```
## [1] "Pour DTW, le cluster 4 a la moyenne de coût DTW la plus faible quand on met le tout les points 
```

```
print(
  paste(
    "Pour SDTW, le cluster",
    which.min(avgAmpAvgClusterSDTW),
    "a la moyenne de coût DTW la plus faible quand on met le tout les points de la query temp à une dis
  )
)
```

```
## [1] "Pour SDTW, le cluster 4 a la moyenne de coût DTW la plus faible quand on met le tout les points
```

```
# Partie 5:

avgAmpClusterDTW <- rep(0, times = nbclusterDTW)
avgAmpClusterSDTW <- rep(0, times = nbclusterSDTW)

for (i in 1:length(reponseViable)) {
  if (queryRef[1] >= fenetresViable[1, i]) {
    avgAmpClusterDTW[resultatPamDTW[i]] <-
```

```
      avgAmpClusterDTW[resultatPamDTW[i]] + dtw_basic(queryRef,
                                                      fenetresViable[, i] + abs(queryRef[1] - fenetresVi
    avgAmpClusterSDTW[resultatPamSDTW[i]] <-
      avgAmpClusterSDTW[resultatPamSDTW[i]] + sdtw(queryRef,
                                                   fenetresViable[, i] + abs(queryRef[1] - fenetresViabl
                                                   gamma = g)
  } else{
    avgAmpClusterDTW[resultatPamDTW[i]] <-
      avgAmpClusterDTW[resultatPamDTW[i]] + dtw_basic(queryRef,
                                                      fenetresViable[, i] - abs(queryRef[1] - fenetresVi
    avgAmpClusterSDTW[resultatPamSDTW[i]] <-
      avgAmpClusterSDTW[resultatPamSDTW[i]] + sdtw(queryRef,
                                                   fenetresViable[, i] - abs(queryRef[1] - fenetresViabl
                                                   gamma = g)
  }
}

for (i in 1:nbclusterDTW) {
  avgAmpClusterDTW[i] <-
    avgAmpClusterDTW[i] / table(resultatPamDTW)[i]
}

for (i in 1:nbclusterSDTW) {
  avgAmpClusterSDTW[i] <-
    avgAmpClusterSDTW[i] / table(resultatPamSDTW)[i]
}

cat("\nPartie 5\n")
```

```
##
## Partie 5
```

```
print(
  paste(
    "Pour DTW, le cluster",
    which.min(avgAmpClusterDTW),
    "a la moyenne de coût DTW la plus faible quand on met le 1er point de la query temp à niveau de la
  )
)
```

```
## [1] "Pour DTW, le cluster 1 a la moyenne de coût DTW la plus faible quand on met le 1er point de la
```

```
print(
  paste(
    "Pour SDTW, le cluster",
    which.min(avgAmpClusterSDTW),
    "a la moyenne de coût DTW la plus faible quand on met le 1er point de la query temp à niveau de la
  )
)
```

```
## [1] "Pour SDTW, le cluster 3 a la moyenne de coût DTW la plus faible quand on met le 1er point de la
```

Imputation

```r
repC1DTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])
repC1SDTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])

repC4DTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])
repC4SDTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])

repC5DTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])
repC5SDTW <-
  data.frame("repRef" = donnee[gapStart:(gapStart + gapTaille - 1)])

for (i in 1:length(reponseViable)) {
  if (resultatPamDTW[i] == which.min(avgClusterDTW)) {
    repC1DTW <- cbind(repC1DTW, reponseViable[, i])
  }
  if (resultatPamSDTW[i] == which.min(avgClusterSDTW)) {
    repC1SDTW <- cbind(repC1SDTW, reponseViable[, i])
  }
  if (resultatPamDTW[i] == which.min(avgAmpAvgClusterDTW)) {
    repC5DTW <- cbind(repC4DTW, reponseViable[, i])
  }
  if (resultatPamSDTW[i] == which.min(avgAmpAvgClusterSDTW)) {
    repC5SDTW <- cbind(repC4SDTW, reponseViable[, i])
  }
  if (resultatPamDTW[i] == which.min(avgAmpClusterDTW)) {
    repC5DTW <- cbind(repC5DTW, reponseViable[, i])
  }
  if (resultatPamSDTW[i] == which.min(avgAmpClusterSDTW)) {
    repC5SDTW <- cbind(repC5SDTW, reponseViable[, i])
  }
}
repC1DTW <- subset(repC1DTW, select = -1)
repC1SDTW <- subset(repC1SDTW, select = -1)
repC4DTW <- subset(repC5DTW, select = -1)
repC4SDTW <- subset(repC5SDTW, select = -1)
repC5DTW <- subset(repC5DTW, select = -1)
repC5SDTW <- subset(repC5SDTW, select = -1)

repC1DTW <- t(repC1DTW)
repC1SDTW <- t(repC1SDTW)
repC4DTW <- t(repC5DTW)
repC4SDTW <- t(repC5SDTW)
repC5DTW <- t(repC5DTW)
repC5SDTW <- t(repC5SDTW)


medRepC1DTW <- vector("numeric", length = 0)
```

```r
medRepC1SDTW <- vector("numeric", length = 0)

medRepC4DTW <- vector("numeric", length = 0)
medRepC4SDTW <- vector("numeric", length = 0)

medRepC5DTW <- vector("numeric", length = 0)
medRepC5SDTW <- vector("numeric", length = 0)

avgRepC1DTW <- vector("numeric", length = 0)
avgRepC1SDTW <- vector("numeric", length = 0)

avgRepC4DTW <- vector("numeric", length = 0)
avgRepC4SDTW <- vector("numeric", length = 0)

avgRepC5DTW <- vector("numeric", length = 0)
avgRepC5SDTW <- vector("numeric", length = 0)


for (i in 1:gapTaille) {
  # Median
  medRepC1DTW <- c(medRepC1DTW, quantile(repC1DTW[, i], 0.5))
  medRepC1SDTW <-
    c(medRepC1SDTW, quantile(repC1SDTW[, i], 0.5))

  medRepC4DTW <- c(medRepC4DTW, quantile(repC5DTW[, i], 0.5))
  medRepC4SDTW <-
    c(medRepC4SDTW, quantile(repC5SDTW[, i], 0.5))

  medRepC5DTW <- c(medRepC5DTW, quantile(repC5DTW[, i], 0.5))
  medRepC5SDTW <-
    c(medRepC5SDTW, quantile(repC5SDTW[, i], 0.5))

  # Average

  avgRepC1DTW <- c(avgRepC1DTW, mean(repC1DTW[, i]))
  avgRepC1SDTW <-
    c(avgRepC1SDTW, mean(repC1SDTW[, i]))

  avgRepC4DTW <- c(avgRepC4DTW, mean(repC5DTW[, i]))
  avgRepC4SDTW <-
    c(avgRepC4SDTW, mean(repC5SDTW[, i]))

  avgRepC5DTW <- c(avgRepC5DTW, mean(repC5DTW[, i]))
  avgRepC5SDTW <-
    c(avgRepC5SDTW, mean(repC5SDTW[, i]))
}

medRepC1DTW <- medRepC1DTW + (donnee[gapStart - 1] - medRepC1DTW[1])
medRepC1SDTW <-
  medRepC1SDTW + (donnee[gapStart - 1] - medRepC1SDTW[1])

medRepC4DTW <- medRepC4DTW + (donnee[gapStart - 1] - medRepC4DTW[1])
medRepC4SDTW <-
```

```
  medRepC4SDTW + (donnee[gapStart - 1] - medRepC4SDTW[1])

medRepC5DTW <- medRepC5DTW + (donnee[gapStart - 1] - medRepC5DTW[1])
medRepC5SDTW <-
  medRepC5SDTW + (donnee[gapStart - 1] - medRepC5SDTW[1])

df <-
  data.frame(
    "index" = 1:length(medRepC1DTW),
    "main" = donnee[gapStart:(gapStart + gapTaille - 1)],
    # Median
    "medC1DTW" = medRepC1DTW,
    "medC1SDTW" = medRepC1SDTW,
    "medC4DTW" = medRepC4DTW,
    "medC4SDTW" = medRepC4SDTW,
    "medC5DTW" = medRepC5DTW,
    "medC5SDTW" = medRepC5SDTW,
    # Average
    "avgC1DTW" = avgRepC1DTW,
    "avgC1SDTW" = avgRepC1SDTW,
    "avgC4DTW" = avgRepC4DTW,
    "avgC4SDTW" = avgRepC4SDTW,
    "avgC5DTW" = avgRepC5DTW,
    "avgC5SDTW" = avgRepC5SDTW,
    #DTWBI
    "DTWBI" = compute.DTWBI_QF(dataModif, 20, 0, queryTaille, verbose = FALSE)[gapStart:(gapStart +
                                                                                        gapTaille -
  )
r <- rep(0, times = 7)

cat("\nDTW entre la réponse de référence et la médiane du cluster DTW selon critére 1\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster DTW selon critére 1
```

```
r[1] <- dtw_basic(df$main, df$medC1DTW)
print(r[1])
```

```
## [1] 9.535748
```

```
cat("\nDTW entre la réponse de référence et la médiane du cluster SDTW selon critére 1\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster SDTW selon critére 1
```

```
r[2] <- dtw_basic(df$main, df$medC1SDTW)
print(r[2])
```

```
## [1] 9.535748
```

```r
cat("\nDTW entre la réponse de référence et la médiane du cluster DTW selon critére 4\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster DTW selon critére 4
```

```r
r[3] <- dtw_basic(df$main, df$medC4DTW)
print(r[3])
```

```
## [1] 8.894263
```

```r
cat("\nDTW entre la réponse de référence et la médiane du cluster SDTW selon critére 4\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster SDTW selon critére 4
```

```r
r[4] <- dtw_basic(df$main, df$medC4SDTW)
print(r[4])
```

```
## [1] 11.41617
```

```r
cat("\nDTW entre la réponse de référence et la médiane du cluster DTW selon critére 5\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster DTW selon critére 5
```

```r
r[5] <- dtw_basic(df$main, df$medC5DTW)
print(r[5])
```

```
## [1] 8.894263
```

```r
cat("\nDTW entre la réponse de référence et la médiane du cluster SDTW selon critére 5\n")
```

```
##
## DTW entre la réponse de référence et la médiane du cluster SDTW selon critére 5
```

```r
r[6] <- dtw_basic(df$main, df$medC5SDTW)
print(r[6])
```

```
## [1] 11.41617
```

```r
cat("\nDTW entre la réponse de référence et DTWBI\n")
```

```
##
## DTW entre la réponse de référence et DTWBI
```

```r
r[7] <- dtw_basic(df$main, df$DTWBI)
print(r[7])
```

```
## [1] 1.905602
```

```r
t <- gather(df, key = variable, value = value, -index)
p <-
  ggplot(data = t, aes(x = index, y = value, color = variable))
p <-
  p + geom_line() + labs(x = "Jour",
                         y = "Valeur",
                         # title = paste("DTW, PAM, cluster", i))
                         title = paste("Résultat du bouchage"))

# Affichage du graphique avec plotly
print(ggplotly(p))
```

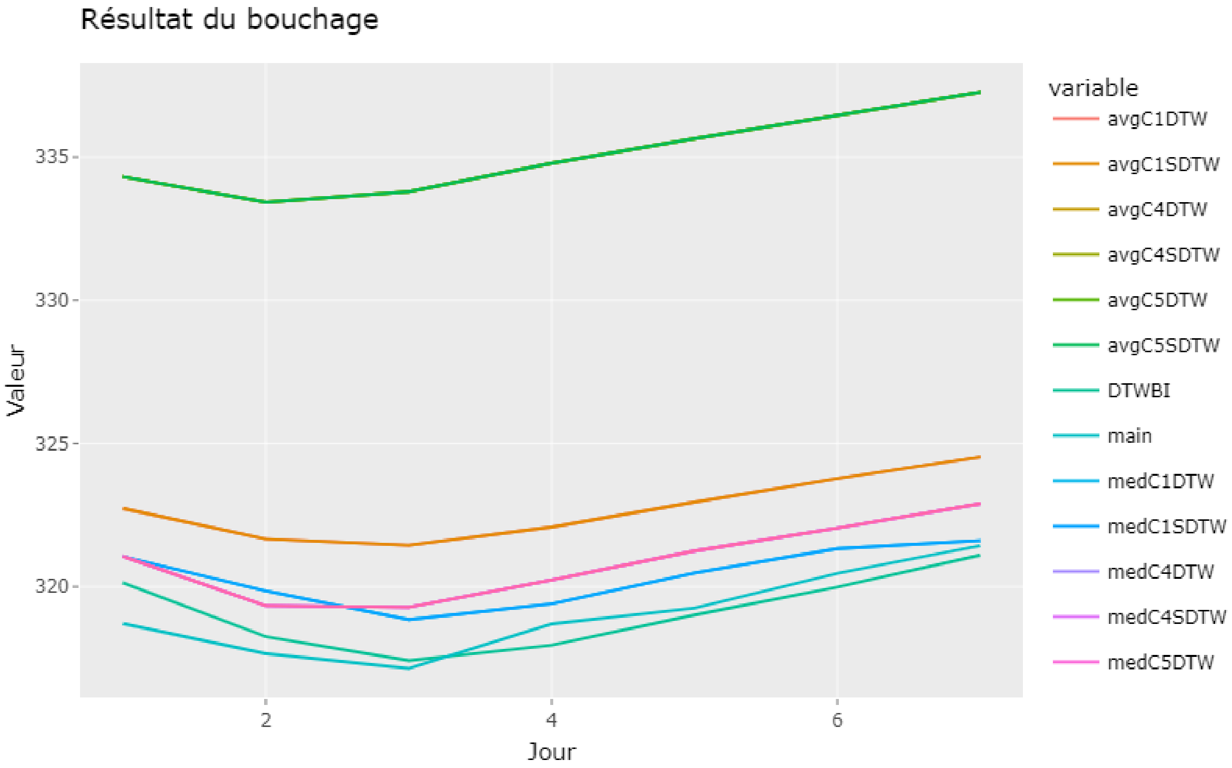


Figure 3: Imputation