Homework 9

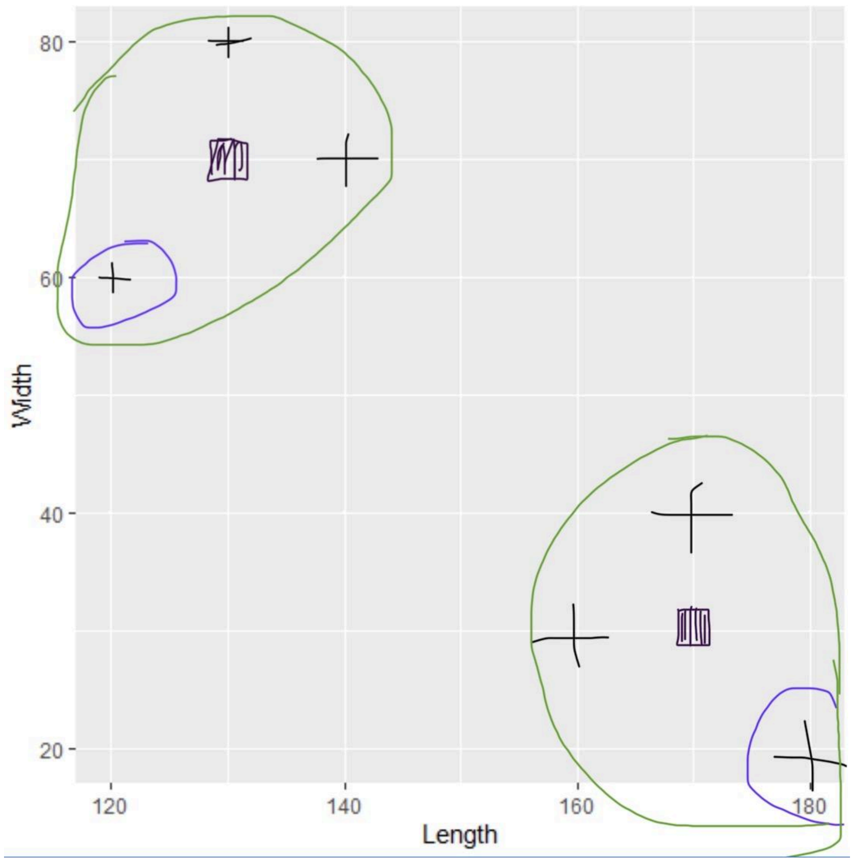
Mariia Nikitash

Prepare your answers as a **single PDF file**.
Group work: You may work in groups of 1-3. Include all group member names in the PDF file. You may work with students in both sections (375-01, -02). Only one person in the group should submit to Canvas.
Due: check on Canvas.

1. Consider the following dataset. (Note: do **NOT** write any code for this problem. The answers are to be computed by hand and marked on the graph. You can even visually guess some of the answers.)

Length	120	140	130	170	160	180
Width	60	70	80	40	30	20

a) Mark the data points on the graph below (*use '+' to indicate each point*).



b) Let k=2. Let one of the two initial centers be (Length=120, Width=60). Select the second center using the **Farthest Distance Heuristic**. Indicate the two centers on the graph (*circle the centers*).

The farthest point is (180,20)

c) Recompute the centers after the first iteration of the k-means algorithm.

1st Center is (120,60) includes (130,80) and (140,70)
The average distance between points in clusters : (120 + 130 + 140) / 3 for Length
And (60+80+70) / 3 for Width. ⇒ (130, 70)

2nd Center is (180, 20) includes (160,30) and (170,40)
The average distance between points in clusters: (170, 30)

New center 1: (130, 70)

New center 2: (170, 30)

Indicate the two new centers on the graph (*mark new centers with squares*).

Presented in the the graph

d) What are the two clusters after this first iteration? *Draw two ovals, each containing all the points in one cluster in the graph above.*

Presented in the the graph

e) Will the k-means algorithm terminate after this first iteration or will it continue? Answer in 1-2 sentences.

The k-means algorithm will continue after the first iteration because convergence hasn't been reached yet. Convergence in k-means is when the centers stabilize, meaning they stop changing significantly between iterations. Since the centers have been updated in the first iteration, further iterations are needed to refine the clusters until convergence is achieved.

f) If a new point (Length=140, Width=60) is given, to which cluster will it belong?
It will go to the 1st cluster since the point is closer to it's center

2. Consider the file `breast-cancer-wisconsin.csv` (in the Datasets module on Canvas) which contains "Features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass."[1] The goal is to cluster the data based on the features to distinguish Benign and Malignant cases.

a. Read the data from the file into an object called "mydata". Column 1 ("Code") is the anonymized subject code and will not be used here. Columns 2-10 are the 9 features.
```
mydata <- read_csv("breast-cancer-wisconsin.csv") %>% select(-`Code`)
```
b. Column 11 is the diagnosis: [B]enign or [M]alignant.
   i. How many total cases are there in the data?: 683
   ```
   nrow(mydata)
   [1] 683
   ```
   ii. How many [B]enign cases are there in the data?: 444
   ```
   sum(mydata$Class == "B")
   [1] 444
   ```
   iii. How many [M]alignant cases are there in the data?: 239
   ```
   sum(mydata$Class == "M")
   [1] 239
   ```
c. Run k-means clustering using **all the rows** and **only the following features**: **ClumpThickness**, **CellSize**, and **Nuclei**. Use nstart=10.
   i. What should be the value of k? k = 2
   Since we have 2 centers the number of clusters should be 2
   ii. Give R code:
   ```
   features <- mydata %>% select(ClumpThickness, CellSize, Nuclei)
   km <- kmeans(features, centers = 2, nstart = 10)
   ```
d. Evaluation: Compare the resulting clusters with the known diagnosis.
   i. Complete the contingency table of your clustering. (Hint: use R's table() function. You can arbitrarily assign cluster 1/2 to Benign/Malignant)

	Cluster 1	Cluster 2
Benign	437	20
Malignant	7	219

   ii. Give R code:
   ```
   cluster_labels <- c("B", "M")[km$cluster]
   contingency_table <- table(cluster_labels, mydata$Class)
   ```

[1]Original dataset from Breast Cancer Wisconsin (Diagnostic) Data Set
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

3. Using the contingency table that you obtained from the previous problem (3.c), calculate the following metrics (consider Malignant as the Positive class):

TP = 219
TN = 437
FP = 7
FN = 20

1. Accuracy = (TP+TN)/All
   656/683=0.9605

2. Error = 1 - Acurracy = 0.0395

3. Precision TP/(TP+FP) = 219/226=0.969

4. Recall TP/P =TP/(TP+FN) =219/239=0.9163

5. F-score = (2 x precision x recall)/(precision x recall) = 0.94191

Consider a "silly" classifier for this problem that makes every prediction as Malignant. Calculate the metrics for this "silly" classifier.

TP = TP + FN = 219 + 20 = 239
TN = 0 (No case is predicted as Benign)
FP = TP (All cases predicted as Malignant) = 239
FN = 0 (No Malignant case is incorrectly predicted as Benign)

1. Accuracy 339+0/683 = 0.49634
2. Error 1 - 0.49634=0.50366
3. Precision TP/(TP+FP) =239/(239+239) = 0.5
4. Recall = TP/P =TP/(TP+FN) = 239/(239+0) = 1
5. F-score = (2 x precision x recall)/(precision x recall) = (2 x 0.5 x 1)/(0.5 + 1) = 0.6667

4. Load the "mystery" vector in file `myvec.RData` on Canvas under Datasets using `load("myvec.RData")`[2]. Decompose the time series data into trend, seasonal, and random components.

Specifically, write R code to do the following:

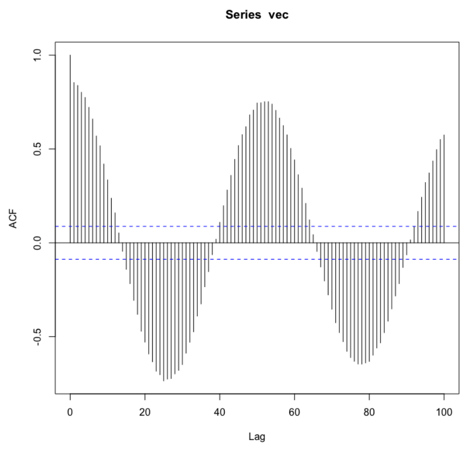
a) Load the data. [show code]

```
load("myvec.RData")
vec <- myvec
```

[2] R allows you to store objects in its own machine-independent binary format, .RData, instead of a text format such as .csv

b) Find the frequency of the seasonal component (Hint: use the autocorrelation plot. You must specify the lag.max parameter in acf() as the default is too small.) [code and plot]

```
acf(vec, lag.max = 100)
```

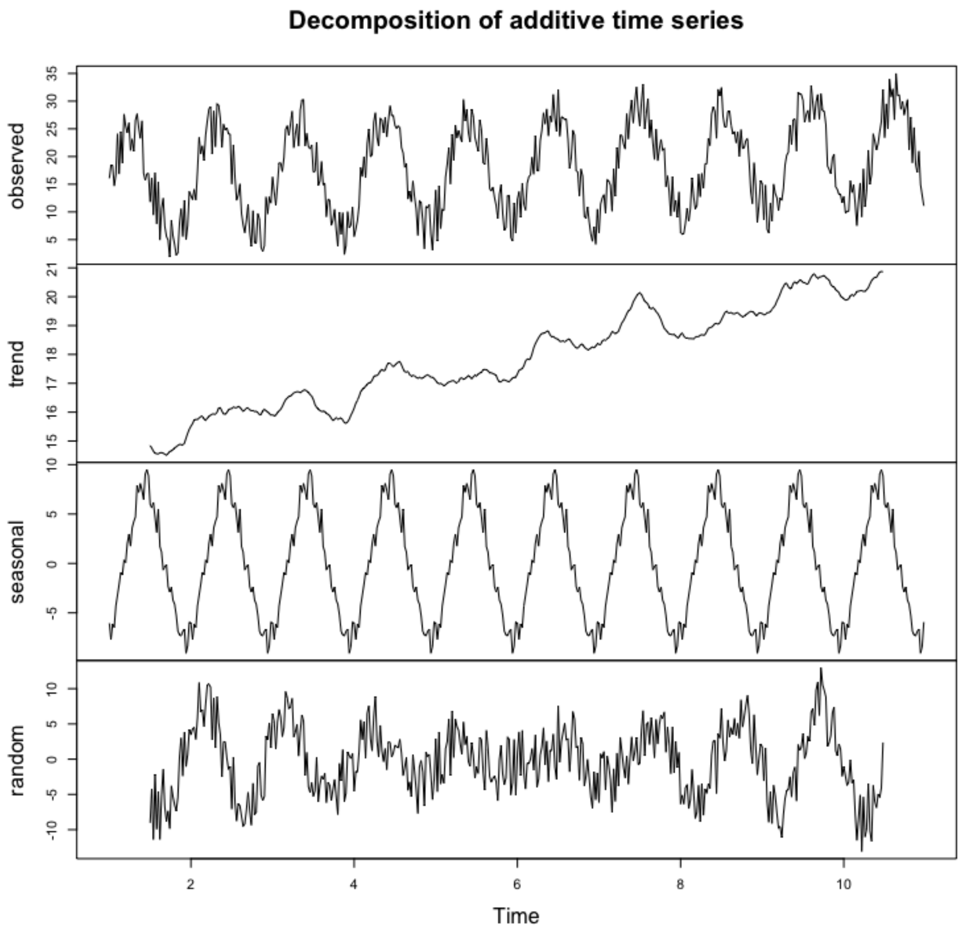


c) Convert to a `ts` object [code]

```
vec.ts <- ts(vec, frequency = 50)
```

d) Decompose the ts object. Plot the output showing the trend, seasonal, random components. [code and plot]

```
decompose(vec.ts)
plot(decompose(vec.ts))
```



5. (Same as classwork problem) Compute the Dynamic Time Warping distance between the two time series, A and B:

A= (2,2), (0,4), (2,6), (4,5)

B= (1,1), (0,6), (4,4)

Use squared Euclidean distance as the cost function:

$cost(A_i, B_j) = (A_{i,1} - B_{j,1})^2 + (A_{i,2} - B_{j,2})^2$.

a) Show the cost matrix. This is partially complete below.

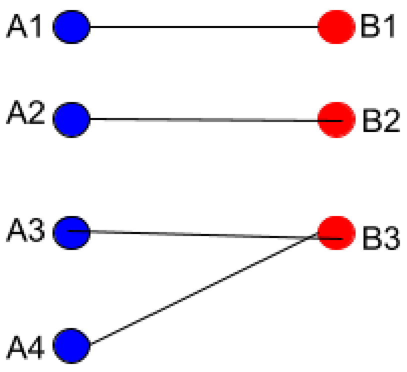
	B_1	B_2	B_3
A_1	2	20	8
A_2	10	4	16
A_3	26	4	8
A_4	25	17	1

b) Show the DTW matrix. This is partially complete below.

2	22	30
12	6	22
38	10	14
63	27	11

c) The DTW distance between the two time-series is 11.

d) Mark the optimal alignment between the two time-series in the diagram below.



6. a) Complete the R function below to compute the DTW distance between two time-series, A and B, each containing 2D points and using the cost function as in the previous question. Note that A and B will have two columns but a varying number of rows.

```
dtw <- function (A, B) {
  M <- nrow(A)
  N <- nrow(B)
  Cost <- matrix(0,M,N) # Initialize with zeros
  for (i in 1:M) {
    for (j in 1:N) {
        Cost[i,j] <- as.numeric((A[i,1] - B[j,1])^2 + (A[i,2] -
B[j,2])^2) # distance function
      }
  }
  C <- matrix(0,M,N) # Initialize with zeros
  C[1,1] <- Cost[1,1] # Value for top left cell
  for (i in 2:M) { # Values for first column
      C[i,1] <- C[i-1,1] + Cost[i,1]
  }
  for (j in 2:N) { # Values for first row
      C[1,j] <- C[1,j-1] + Cost[1,j]
  }
  #
  # Values for other rows and columns
  # TO BE COMPLETED

for (i in 2:M) {
    for (j in 2:N) {
      C[i, j] <- Cost[i, j] + min(C[i - 1, j], C[i, j - 1], C[i - 1, j -
1])
    }
  }

  return (C[M,N])
}
```

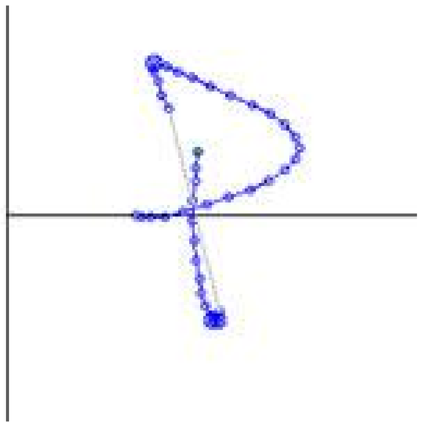
b) Verify your answer to Q5 using the above function. You can create the two input time-series as a two-column data.frame/tibble like so:

```
A <- tibble("x" = c(2, 0, 2, 4), "y" = c(2, 4, 6, 5))
```

[show code and output]

```
A <- tibble("x" = c(2, 0, 2, 4), "y" = c(2, 4, 6, 5))
B <- tibble("x" = c(1,0,4), "y" = c(1,6,4))
dtw(A,B)
[1] 11
```

7. You are given 4 time-series of 2D points (2 column tables) in CSV files: char1_A.csv, char1_E.csv, char1_M.csv, char1_O.csv, and char4_.csv (under Datasets module on Canvas). Each represents one of the English alphabet characters A, E, M, and O as written on a tablet computer[3]. For instance, char1_A.csv, represents the character “A”. Your goal is to identify which character is represented by the 5th file (char4_.csv) using DTW and the cost function used in Q5 and Q6.



a) Explain your approach in 2-3 sentences.

I load all 4 datasets as well as 5th file to inditify whose distance is the closest. After I used the function I created in the previous question to compare each of those 4 files with 5th file (char4_.csv).

b) Show your R code and output

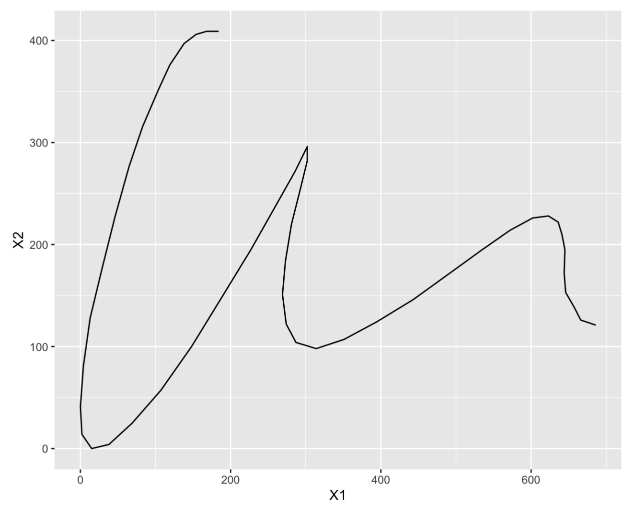
```
CharA <- read_csv("char1_A.csv")
```

[3] Data from UCI Machine Learning repository: https://archive.ics.uci.edu/ml/datasets/UJI+Pen+Characters

```
CharE <- read_csv("char1_E.csv")
CharM <- read_csv("char1_M.csv")
CharO <- read_csv("char1_O.csv")
Char4 <- read_csv("char4_.csv")

dtw(CharA, Char4)
[1] 3562468

dtw(CharE, Char4)
[1] 2335604

dtw(CharM, Char4)
[1] 979030

dtw(CharO, Char4)
[1] 2089999
```

c) char4_.csv represents character: M

```
ggplot( data = CharM) + geom_path(aes(x=X1, y=X2))
```



Hint: Use the DTW function from Q6. To check your answer, you can visualize the series of 2D points using geom_path().