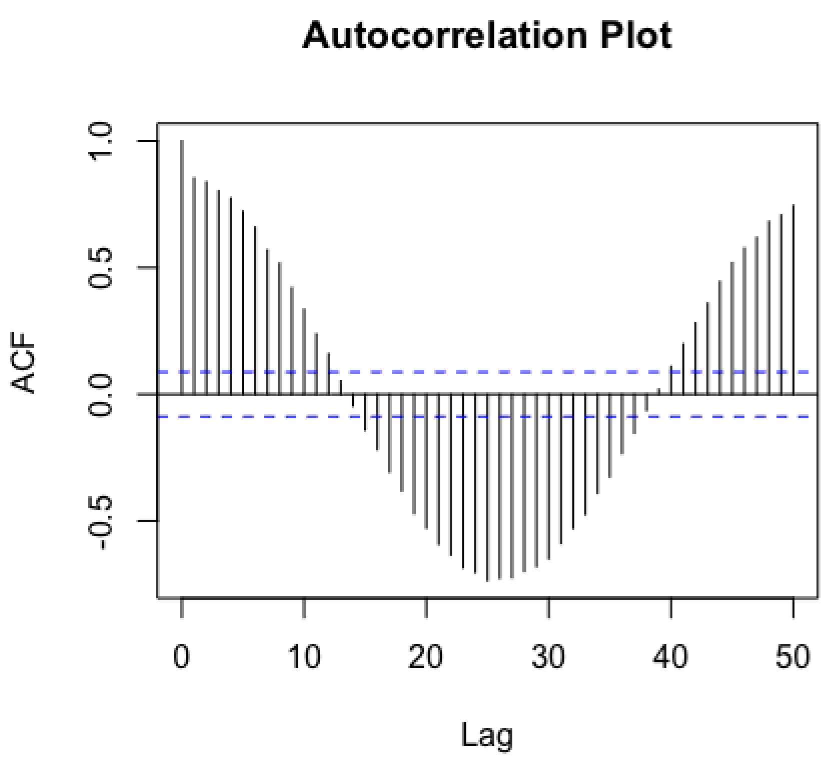
Homework 8

1. Load the "mystery" vector in file `myvec.RData` on Canvas under Datasets using `load("myvec.RData")`[1]. Decompose the time series data into trend, seasonal, and random components.
Specifically, write R code to do the following:

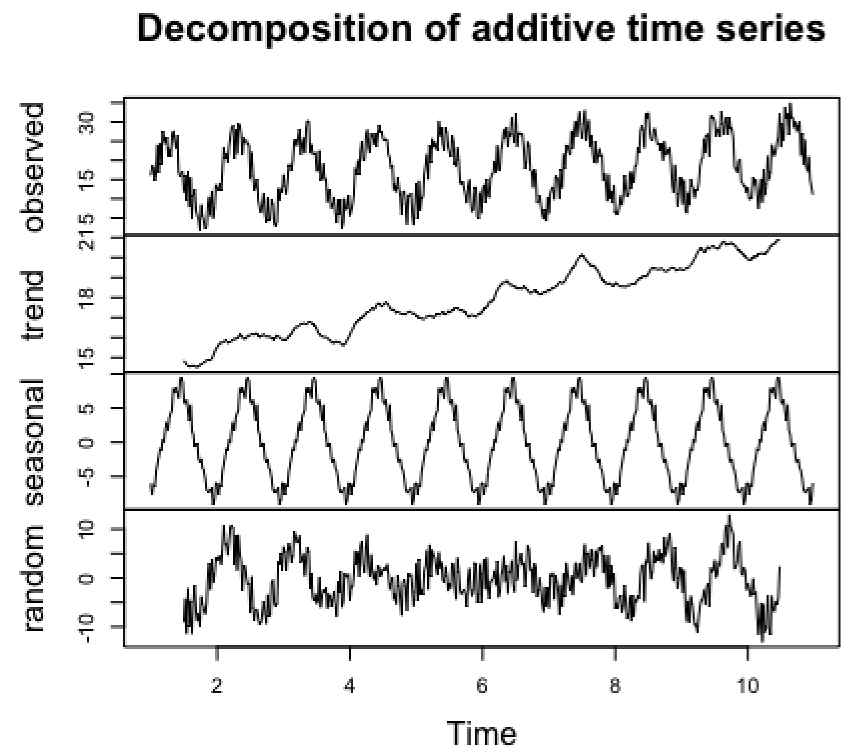
a) Load the data. [show code]
   load("/Users/andresquintana/Downloads/myvec.RData")
b) Find the frequency of the seasonal component (Hint: use the autocorrelation plot. You must specify the lag.max parameter in acf() as the default is too small.) [code and plot]
   acf(myvec, lag.max = 50, main = "Autocorrelation Plot")



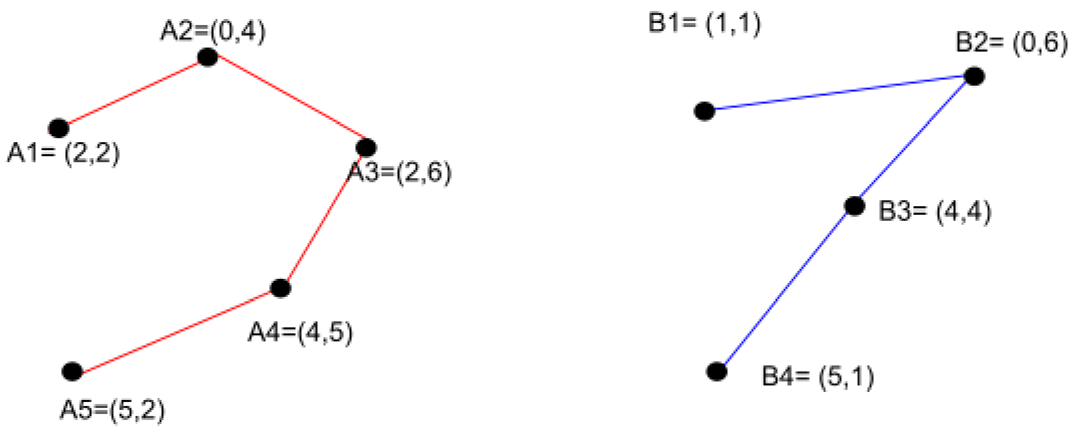
c) Convert to a `ts` object [code]
   mystery_ts <- ts(myvec, frequency = 50)
d) Decompose the ts object. Plot the output showing the trend, seasonal, random components. [code and plot]
   plot(decompose(mystery_ts))

---

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



2. Compute the Dynamic Time Warping distance between the two time series, A and B:
A= (2,2), (0,4), (2,6), (4,5), (5,2)
B= (1,1), (0,6), (4,4), (5,1)



Use squared Euclidean distance as the cost function

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

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

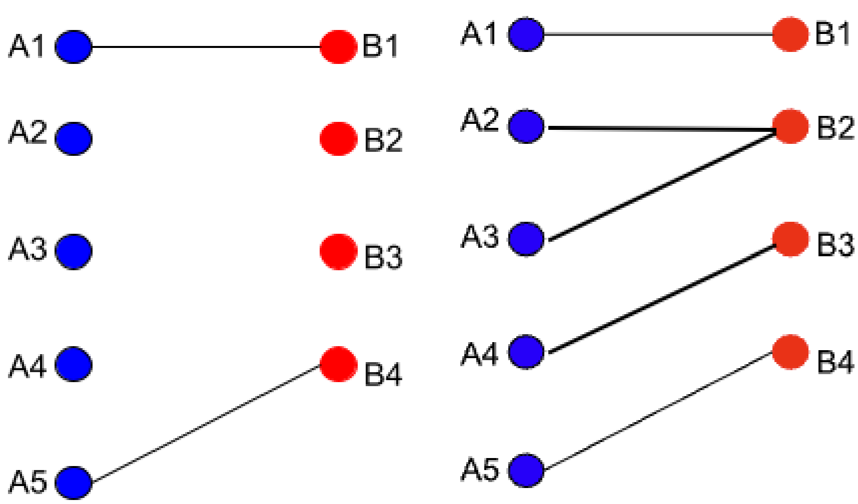
	B_1	B_2	B_3	B_4
A_1	2	20	8	10
A_2	10	4	16	34
A_3	26	4	8	34
A_4	25	17	1	17
A_5	17	41	5	1

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

2	22	30	40
12	6	22	56
38	10	14	48
63	27	**11**	**28**
80	68	**16**	**12**

c) The DTW distance between the two time-series is 12.

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



3. 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 Q2 above. So 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]
  }
  for (i in 2:M) { # Values for other rows and columns
      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 Q2 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))
```

```
> dtw <- function(A, B) {
+     M <- nrow(A)
+     N <- nrow(B)
+     Cost <- matrix(0, M, N)  # Initialize with zeros
+
+     # Fill in the cost matrix
+     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
+
+     # Fill in the values for the first column
+     for (i in 2:M) {
+         C[i, 1] <- C[i - 1, 1] + Cost[i, 1]
+     }
```

```
+
+    # Fill in the values for the first row
+    for (j in 2:N) {
+       C[1, j] <- C[1, j - 1] + Cost[1, j]
+    }
+
+    # Fill in the rest of the matrix
+    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])
+ }
> A <- tibble("x" = c(2, 0, 2, 4, 5), "y" = c(2, 4, 6, 5, 2))
> B <- tibble("x" = c(1, 0, 4, 5), "y" = c(1, 6, 4, 1))
> dtw(A, B)
[1] 12
```

4. You are given 5 time-series of 2D points (2 column tables) in CSV files: ts2.csv, ts3.csv, ts4.csv, ts5.csv, and tsX.csv (in Datasets module on Canvas). Your goal is to identify which of the time series, ts2-ts5, is most similar to the tsX time series using DTW.

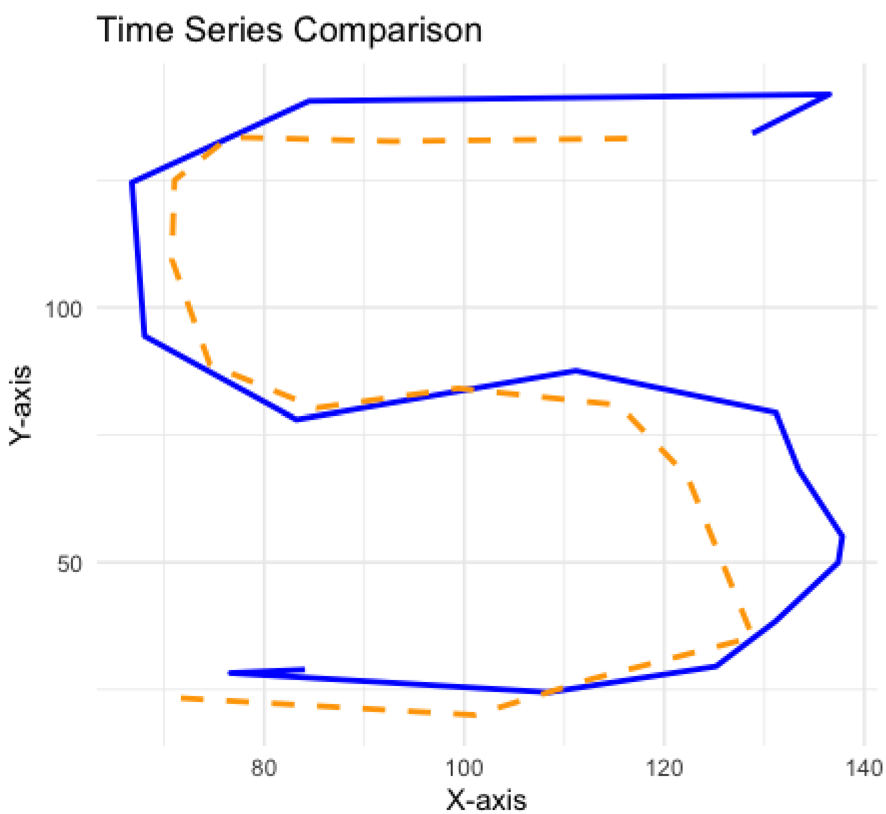
a) Explain your approach in 2-3 sentences.

In order to find the time series most similar to tsX using DTW, I would iterate through ts2-ts5, compute the DTW distance between each of them and tsX, and then identify the one with the smallest DTW distance.

b) Show your R code

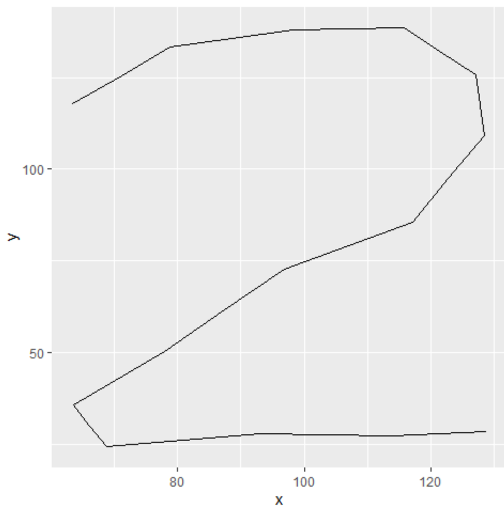
```
tsX <- read_csv("Downloads/tsX.csv")
ts2 <- read_csv("Downloads/ts2.csv")
ts3 <- read_csv("Downloads/ts3.csv")
ts4 <- read_csv("Downloads/ts4.csv")
ts5 <- read_csv("Downloads/ts5.csv")

dtw(as_tibble(ts2), as_tibble(tsX))
[1] 44116.78
dtw(as_tibble(ts3), as_tibble(tsX))
[1] 18583.75
dtw(as_tibble(ts4), as_tibble(tsX))
[1] 13293.01
dtw(as_tibble(ts5), as_tibble(tsX))
[1] 3192.354
```

```
ggplot() +
+     geom_path(data = as_tibble(tsX), aes(x = x, y = y), color = "blue", size = 1) +
+     geom_path(data = as_tibble(ts5), aes(x = x, y = y), color = "orange", size = 1, linetype =
"dashed") +
+     labs(title = "Time Series Comparison",
+          x = "X-axis",
+          y = "Y-axis") +
+     theme_minimal()
>
```



c) tsX is most similar to: ts5

Hint: Use the DTW function from Q3. You can visualize the series of 2D points using geom_path(). For example, ts2:


100
50
y
80
100
120
x