Classifying Exercise with DTW

Liam Hayes

Problem Definition

Given signals from an exercise, can we classify which exercise it is? In this case we know that an exercise is being performed, we just want to be able to classify which exercise it is.

The subject is wearing 5 sensors, each with an accelerometer, gyroscope, and magnemometer. Each accelerometer, gyroscope, and magnemometer tracks movement in the x, y, and z planes. This means for each subject we have $5 \times 3 \times 3 = 45$ 1-dimensional signals to compare.

Method

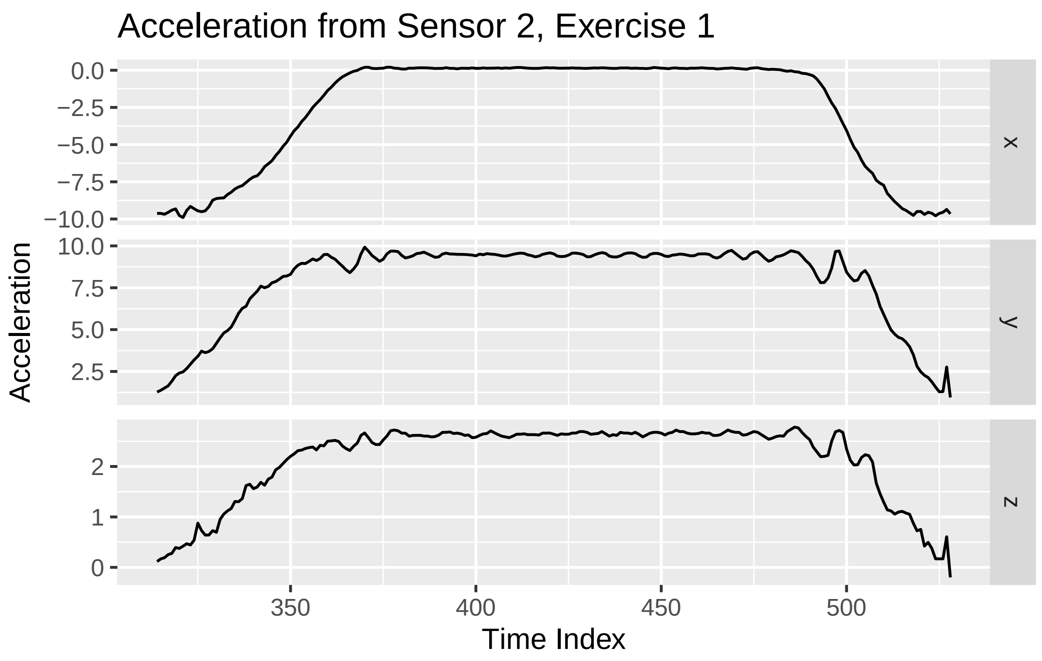
From the data we have 5 subjects performing 8 different exercises. Each subject performs an exercise slightly differently, so to account for this we take the average signal over all subjects for each exercise. We will then take the individual exercise we want to classify and compare it to each of the 8 average exercise signals.

1. Average each signal over all subjects for each exercise
2. Compare the exercise we want to classify to each of the average exercise signals

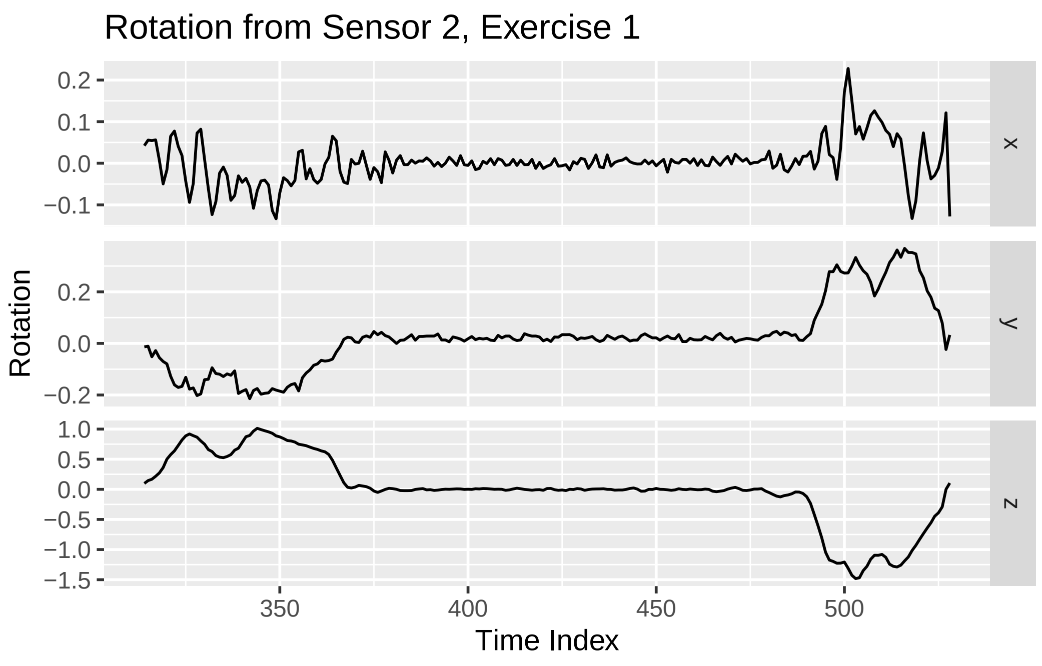
The method of comparison will be Dynamic Time Warping. This allows us to avoid 1-to-1 comparison of the indexes of the two signals. This way, we can compare signals while accounting for different length, speed, and amplitude.

Details

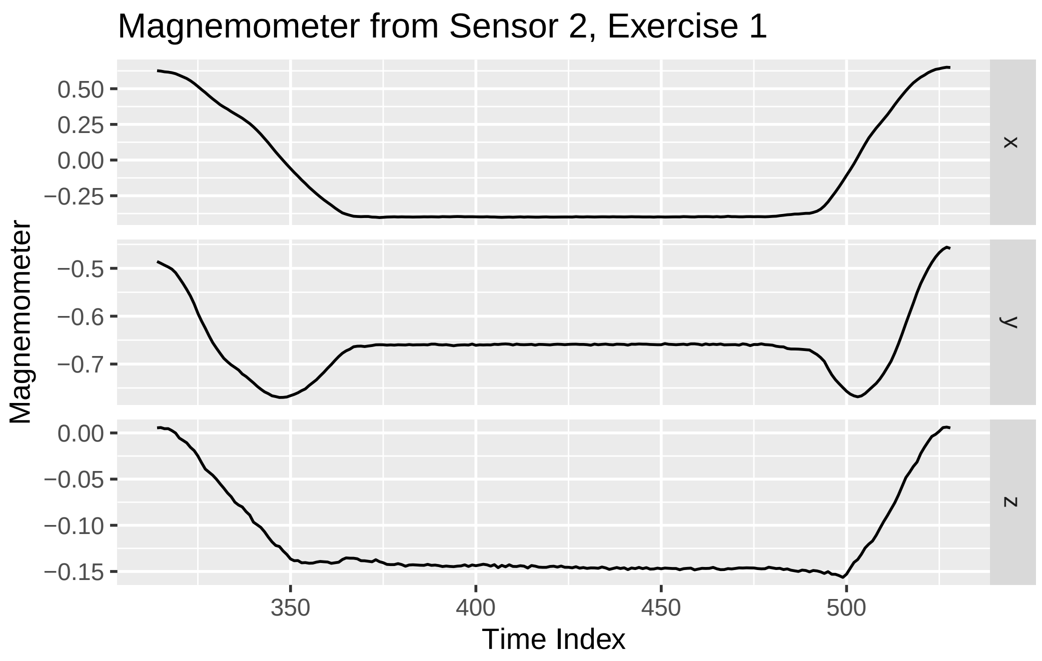
Let's start with looking at data from a single sensor. These plots show one rep of an exercise from sensor 2.



Acceleration from Sensor 2, Exercise 1
Acceleration
x
y
z
Time Index

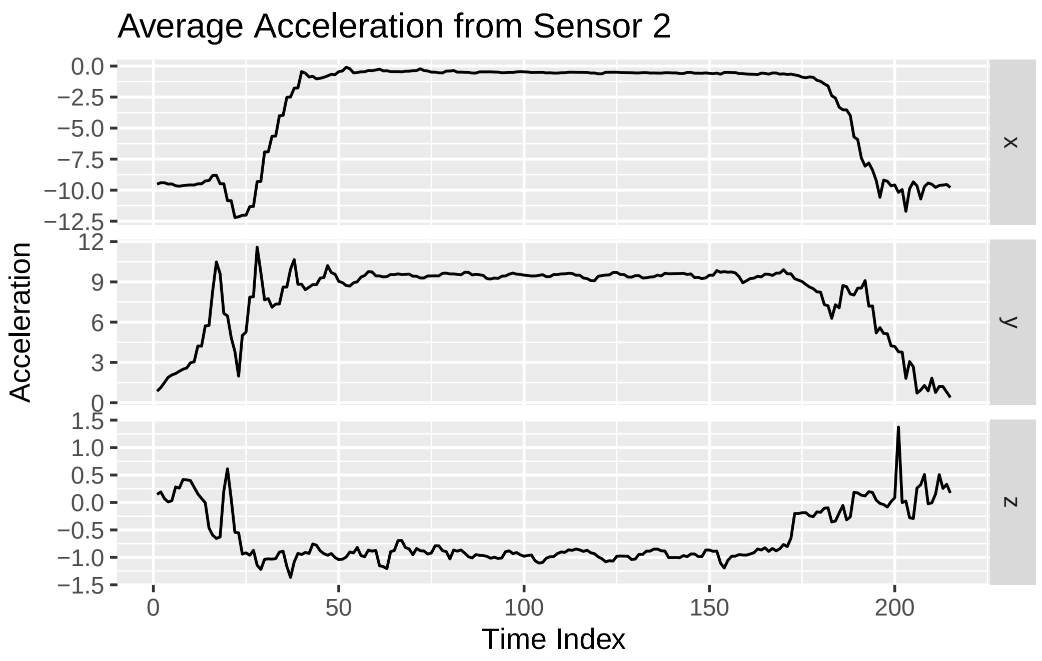


Rotation from Sensor 2, Exercise 1
Rotation
x
y
z
Time Index

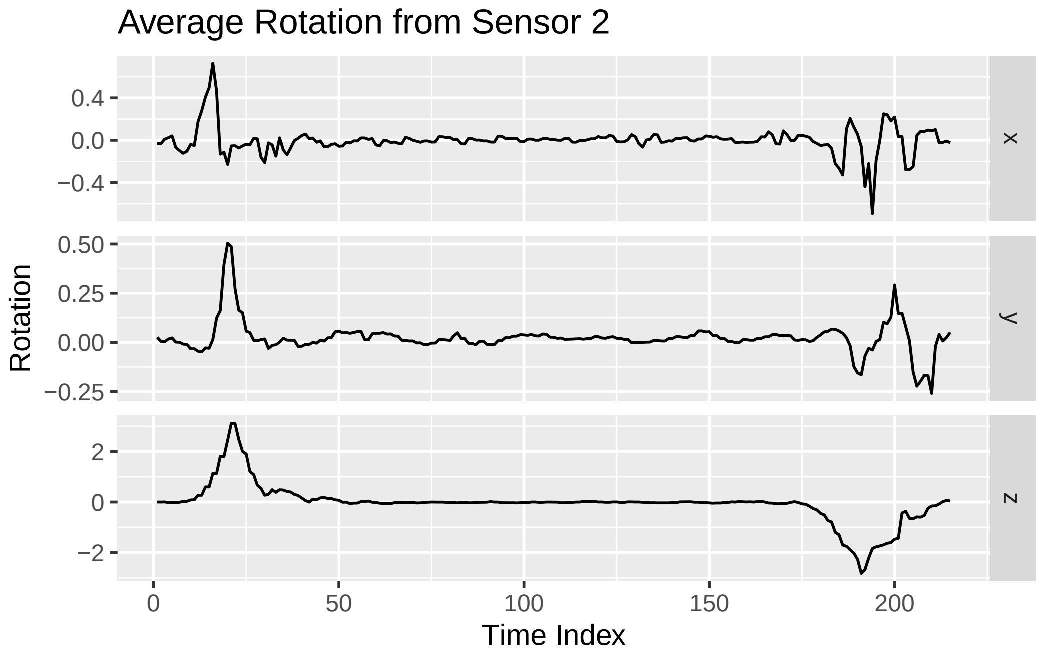


We want to find what an average exercise looks like. To do this, we can use Dynamic Time Warping to match signals of the same type and take the average of those over all sessions. By signals of the same type, I mean we average the x coordinate of acceleration from sensor 1 over all sessions, the x coordinate of rotation (from the gyroscope) from sensor 1 over all sessions, and so on. Since we have 5 sessions, 8 exercises in each session, 5 sensors for each exercise, 3 metrics from each sensor, and 3 planes for each metric, this gets pretty complicated. See functions avgExercise and getAllAvgExercises in the appendix to see how we did this. We then end up with a list of 8 average exercises, each with 45 signals that describe that exercise.

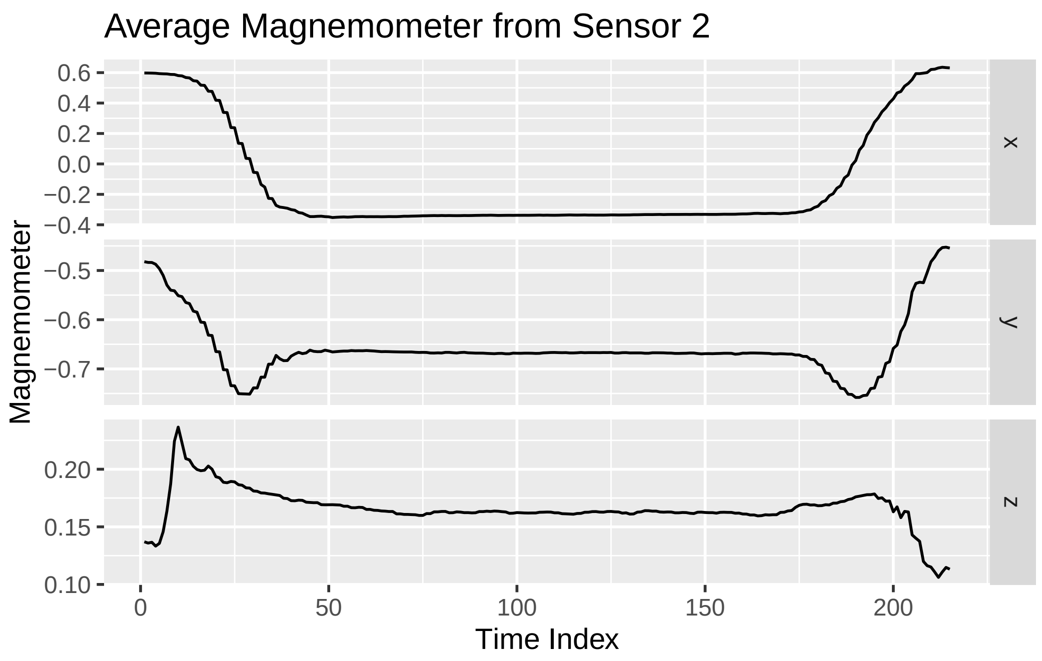
Here are plots of the same exercise in the previous example, averaged over all of the sessions:



Average Acceleration from Sensor 2
Acceleration
Time Index
x
y
z



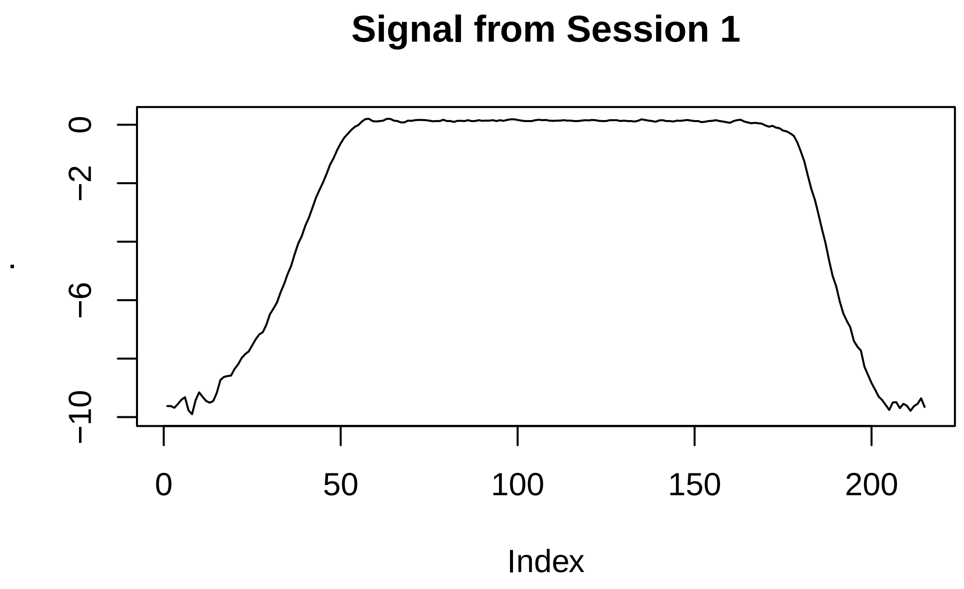
Average Rotation from Sensor 2
Rotation
Time Index
x
y
z



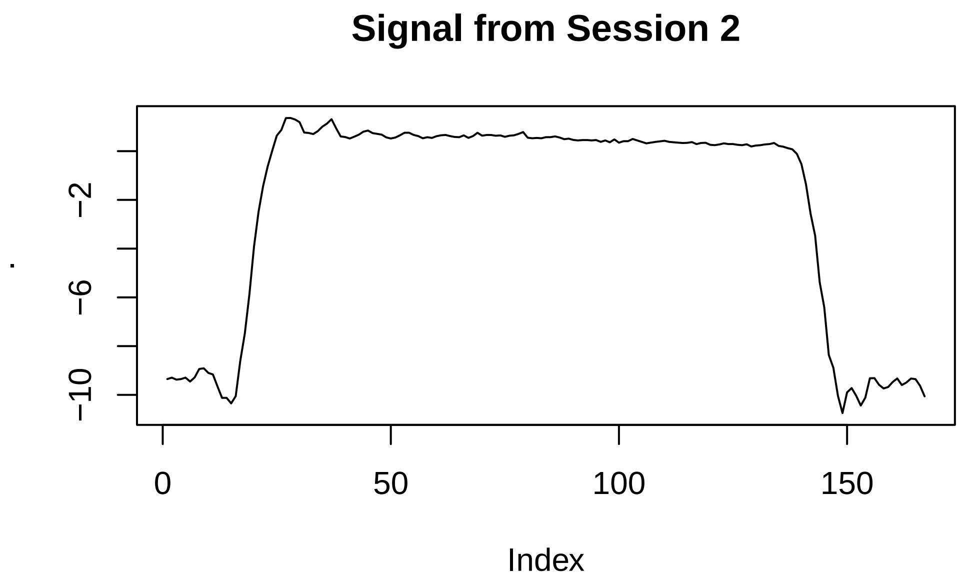
Dynamic Time Warping

Let's take a quick look at Dynamic Time Warping to see the mechanism behind this comparison. Dynamic Time Warping is an algorithm that takes two 1-dimensional signals and "dynamically warps" the time indexes so the signals match up. It also calculates the distance between the two signals to see how different they are.

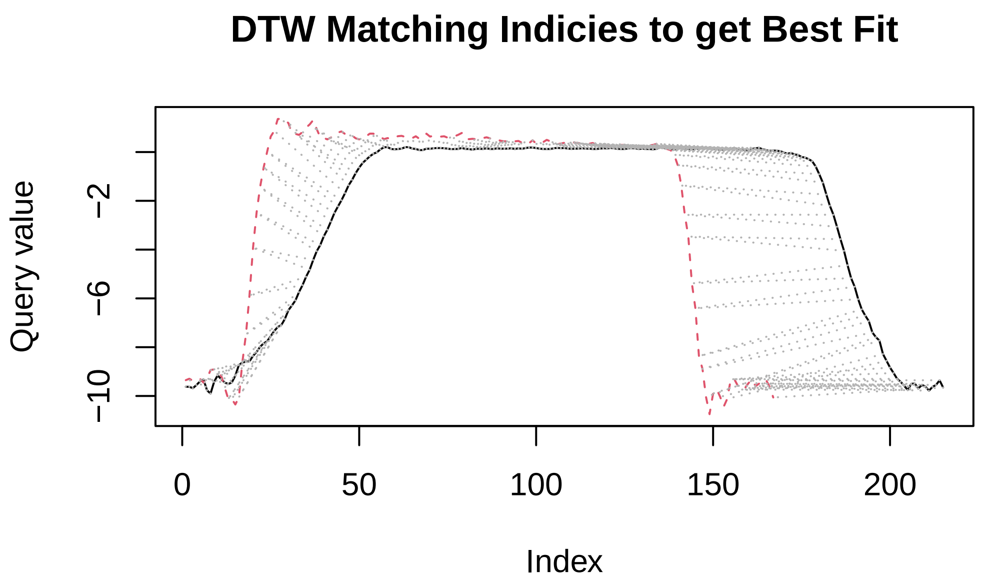
For this example, we'll take the x coordinate of acceleration from exercise 1 and sensor 2 and compare two different sessions.



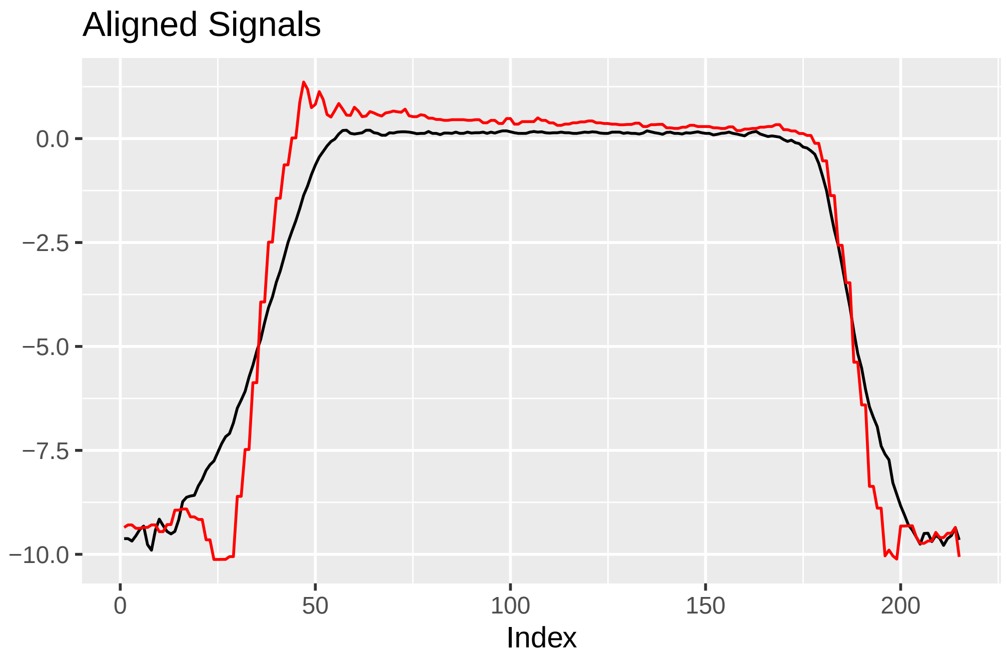
Signal from Session 1
0
−2
−6
−10
0
50
100
150
200
Index



Signal from Session 2
−2
−6
−10
0
50
100
150
Index

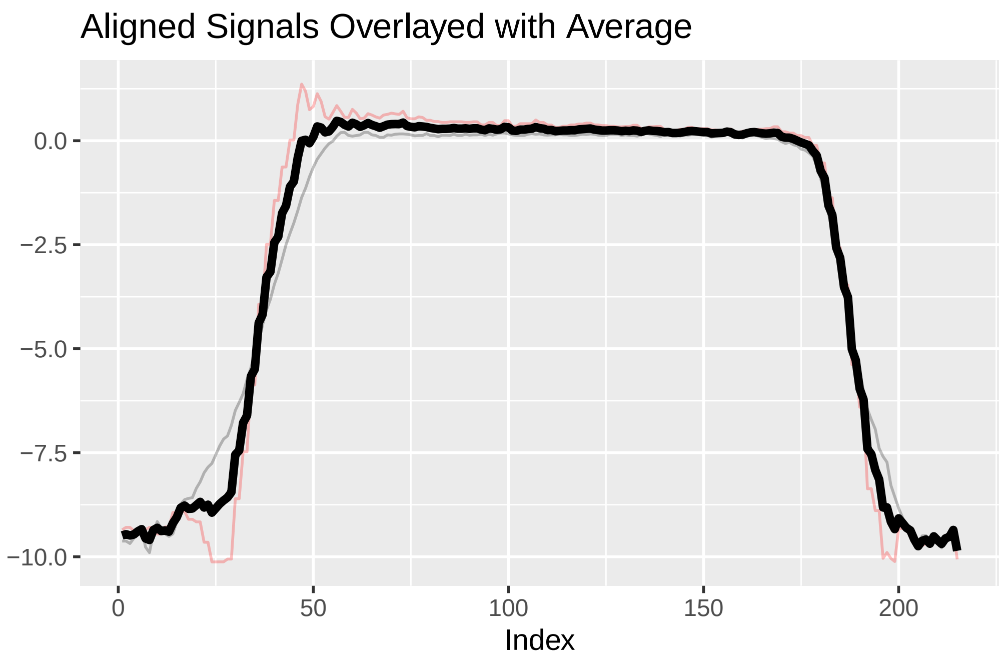


We can then stretch one signal to be the same length as the other signal:



This algorithm also gives us a distance measure to see how different these signals are. In this case the distance is 125.046 .

Finally we can get the average of the two signals:



Classify Exercises

To classify exercises, we compute the distance between the exercise in question and the 8 average exercises that we derived previously. To do this, we need compute the distance between the 45 signals that describe the average exercise and the 45 signals that describe the unclassified exercise and take the average of those 45 distances. This gives us an overall distance measure for each of the 8 exercises, and we choose the lowest distance for our chosen classification.

```
[1] "Exercise 1 distance: 499.067789514144"
[1] "Exercise 2 distance: 460.955220748598"
[1] "Exercise 3 distance: 328.435698508271"
[1] "Exercise 4 distance: 337.658247534729"
[1] "Exercise 5 distance: 326.714324837249"
[1] "Exercise 6 distance: 85.3379060975744"
[1] "Exercise 7 distance: 220.840223938658"
[1] "Exercise 8 distance: 462.797135207845"
```

```
[1] "Classification: 6"
```

In this case we choose exercise 6 with a distance of 85.338, which is the correct classification.

Code

```r
# Set working directory and load packages
setwd('C:/Users/lchco/OneDrive/Documents/School/CSU/Spring_2024/STAT472/physical+therapy+e

library(dtw)
library(tidyverse)
library(ggplot2)

# Extracts a signle rep of a certain type from the data
getRep <- function(template_session, template_time, type) {
  t <- template_time %>%
    filter(execution.type == type)
  template_session %>%
    filter(time.index >= t$start & time.index <= t$end)
}

# Average over all subjects for a specific exercise, sensor, and exercise style
avgExercise <- function(exercise, sensor, exerciseStyle) {
  e <- exercise
  u <- sensor
  session_list <- list()
  for (i in 1:5) {
    df <- read.table(file=paste('s',i,'/e',e,'/u',u,'/template_session.txt', sep=''), head
      getRep(read.table(file=paste('s',i,'/e',e,'/template_times.txt', sep=''), header=TRU
      select(-time.index)
    colnames(df) <- paste(colnames(df), i, sep='_')
    session_list <- c(session_list, df)
  }

  df <- list()
  for (i in 1:9) {
    metrics <- session_list[seq(i,45,by=9)]
    m <- which(lengths(metrics)==max(lengths(metrics)))[1]
    vec1 <- metrics[[m]]
    metrics <- metrics[-m]
    while (length(metrics) >= 1) {
      m <- which(lengths(metrics)==max(lengths(metrics)))[1]
      vec2 <- metrics[[m]]
      metrics <- metrics[-m]
      alignment <- tryCatch(
        {
          dtw(vec1, vec2, k=T, step=typeIIIc)
        },
        error = function(e) {
          vec2 <- c(0,vec2,0)
          dtw(vec1, vec2, k=T, step=typeIIIc)
        }
      )
      new_vec2 <- rep(0, length(alignment$index2))
      new_vec2[alignment$index1] <- vec2[alignment$index2]

      vec1 <- (vec1 + new_vec2)/2
    }
    df <- c(df, list(vec1))
  }
  names(df) <- c(paste('acc_x_e',e,'_u',u,sep=''),
                 paste('acc_y_e',e,'_u',u,sep=''),
                 paste('acc_z_e',e,'_u',u,sep=''),
                 paste('gyr_x_e',e,'_u',u,sep=''),
                 paste('gyr_y_e',e,'_u',u,sep=''),
                 paste('gyr_z_e',e,'_u',u,sep=''),
                 paste('mag_x_e',e,'_u',u,sep=''),
                 paste('mag_y_e',e,'_u',u,sep=''),
                 paste('mag_z_e',e,'_u',u,sep=''))
  df %>% as.data.frame() %>%
    mutate(time.index = 1:length(vec1))
}

# Combine the averages in a dataframe
getAllAvgExercises <- function(exerciseType) {
  es <- exerciseType
  exercises <- list()
  for (e in 1:8) {
    sensors <- list()
    for (s in 1:5) {
      sen <- avgExercise(e, s, es)
      sensors <- c(sensors, sen)
    }
    sensors <- sensors %>%
      as.data.frame() %>%
      select(-c(time.index.1, time.index.2, time.index.3, time.index.4)) %>%
      relocate(time.index)
    exercises <- c(exercises, list(sensors))
  }
  exercises
}

# Extract an exercise to compare to the averages
getExercise <- function(exercise, session, startTime, endTime) {
  s <- session
  e <- exercise
  sensors <- list()
  for (u in 1:5) {
    tSesh <- read.table(file=paste('s',s,'/e',e,'/u',u,'/template_session.txt', sep=''), h
    index <- tSesh$time.index
    tSesh <- tSesh %>%
      select(-time.index)
    names(tSesh) <- paste(names(tSesh),'_e',e,'_u',u,sep='')
    sensors <- c(sensors, tSesh)
  }

  sensors <- sensors %>%
    as.data.frame() %>%
    mutate(time.index=index) %>%
    relocate(time.index)

  sensors %>%
    filter(time.index >= startTime & time.index <= endTime)
}

# Compare an exercise to an average
compareExercises <- function(avgExercise, exercise) {
  distances <- rep(0,45)
  for (i in 2:46) {
    distances[i-1] <- dtw(avgExercise[,i], exercise[,i], k=T, step=typeIIIc)$distance
  }
  mean(distances)
}

avgExercise_type1 <- getAllAvgExercises(1)
tTime <- read.table(file=paste('s',1,'/e',1,'/template_times.txt', sep=''), header=TRUE, s
```

```r
compare_exercise <- 6 # This is the exercise that we want to classify

distances <- rep(0, 8)
for (e in 1:8) {
  d <- compareExercises(avgExercise_type1[[e]],
                   getExercise(exercise=compare_exercise,
                               session=1,
                               startTime=tTime$start[1],
                               endTime=tTime$end[1]))
  distances[e] <- d
  print(paste("Exercise ",e," distance: ", d, sep=''))
}

print(paste("Classification: ", which.min(distances), sep=''))
```