exercise_3

Gioele Pinana

2022-05-06

Task and Inputs

In this task we are using the method proposed in Laube and Purves (2011) about segmenting trajectories. The task will be to understand this implementation and apply it on Caro, with a different sampling interval.

libraries

```
library(readr)
library(ggplot2)
library(dplyr)
library(SimilarityMeasures)
library(tidyr)
```

Task 1: Segmentation

Importing dataset

```
caro <- read_csv("caro60.csv")
```

```
## Rows: 200 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): TierID, TierName
## dbl  (3): CollarID, E, N
## dttm (1): DatetimeUTC
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The sampling interval for this dataset is 1 minute. We use a temporal window of 6 minutes. We need to calculate the following Euclidean distances (pos representing single location):

1. pos[n-3] to pos[n]
2. pos[n-2] to pos[n]
3. pos[n-1] to pos[n]
4. pos[n] to pos[n+1]
5. pos[n] to pos[n+2]

6. pos[n] to pos[n+3]

```
caro2 <- caro %>%
  mutate(
    nMinus3 = sqrt((lag(E,3)-E)^2+(lag(N,3)-N)^2),   # distance to pos -3 minutes
    nMinus2 = sqrt((lag(E,2)-E)^2+(lag(N,2)-N)^2),   # distance to pos -2 minutes
    nMinus1 = sqrt((lag(E,1)-E)^2+(lag(N,1)-N)^2),   # distance to pos -1 minutes
    nPlus1  = sqrt((E-lead(E,1))^2+(N-lead(N,1))^2), # distance to pos +1 mintues
    nPlus2  = sqrt((E-lead(E,2))^2+(N-lead(N,2))^2), # distance to pos +2 minutes
    nPlus3  = sqrt((E-lead(E,3))^2+(N-lead(N,3))^2)  # distance to pos +3 minutes
  )
```

Task 2: Specify and apply threshold d

```
caro3 <- caro2 %>%
  rowwise() %>%
  mutate(
    stepMean = mean(c(nMinus3, nMinus2, nMinus1,nPlus1,nPlus2, nPlus3))
  ) %>%
  ungroup()
```

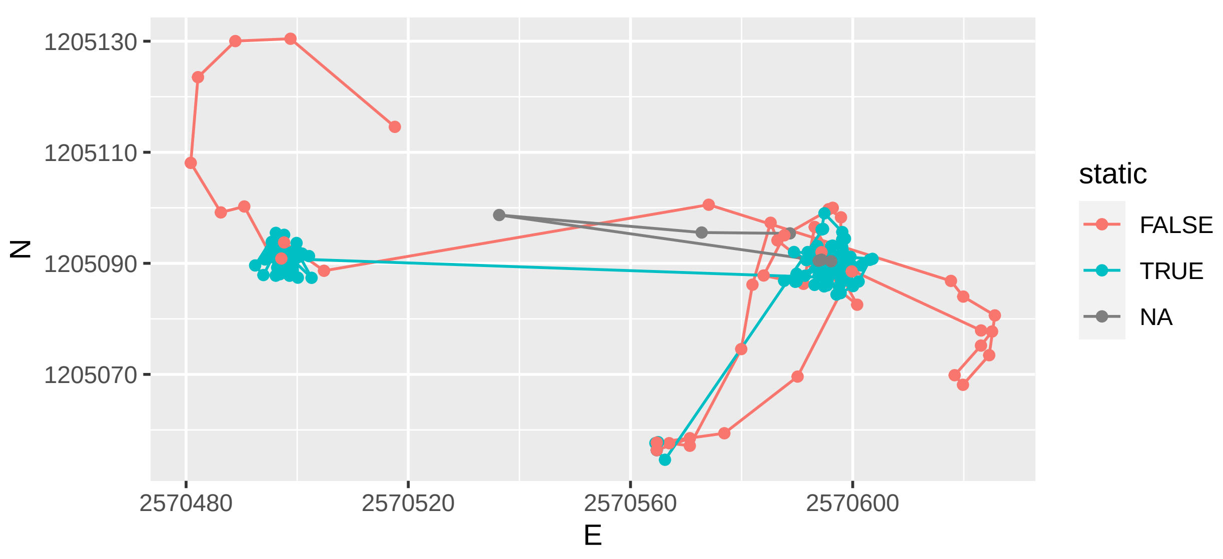
In the last code line I calculated the mean distance of n Minus 3, nMinus2, nMinus1, nPlus1, nPlus2 and nPlus3 for each row. Now I can determine if an animal is moving or not by specifying a threshold distance on stepMean. For this task I use the mean value as a threshold: Positions with distances below this value are considered static

```
caro4 <- caro3 %>%
  ungroup() %>%
  mutate(static = stepMean < mean(stepMean, na.rm = TRUE))

caro_filter <- caro4 %>%
  filter(!static)
```

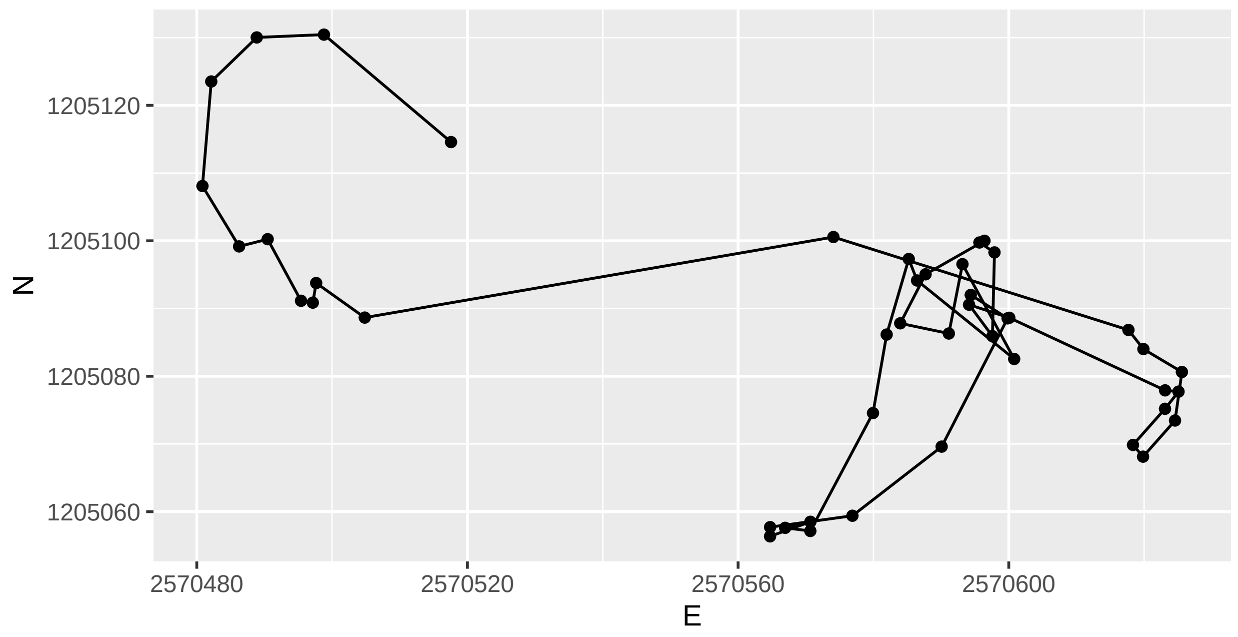
Task 3: Visualize segmented trajectories

Now I visualize the segmented trajectory spatially

```
ggplot(caro4, aes(E, N, colour = static)) +
  geom_path() +
  geom_point() +
  coord_equal()
```



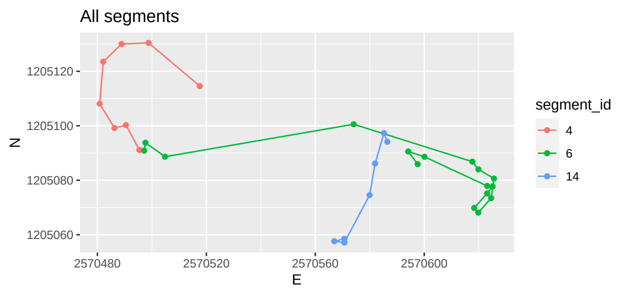
```
ggplot(caro_filter, aes(E, N)) +
  geom_path() +
  geom_point() +
  coord_equal()
```

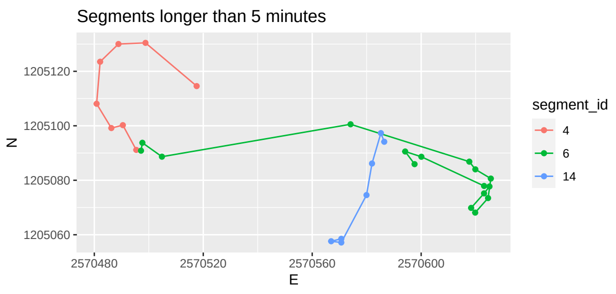


Task 4: Segment-based analysis

```
rle_id <- function(vec){
  x <- rle(vec)$lengths
  as.factor(rep(seq_along(x), times=x))
}

caro5 <- caro4 %>%
  mutate(segment_id = rle_id(static))%>%
  filter(!static) %>%
  group_by(segment_id) %>%
  mutate(duration = n()) %>%
  filter(duration>6)
```

```
par(mfrow=c(1,2))

ggplot(caro5, aes(E, N, colour = segment_id)) +
  geom_path() +
  geom_point() +
  coord_equal() +
  labs(title = "All segments")

ggplot(caro5, aes(E, N, colour = segment_id)) +
  geom_path() +
  geom_point() +
  coord_equal() +
  labs(title = "Segments longer than 5 minutes")
```





Task 5: Similarity measures

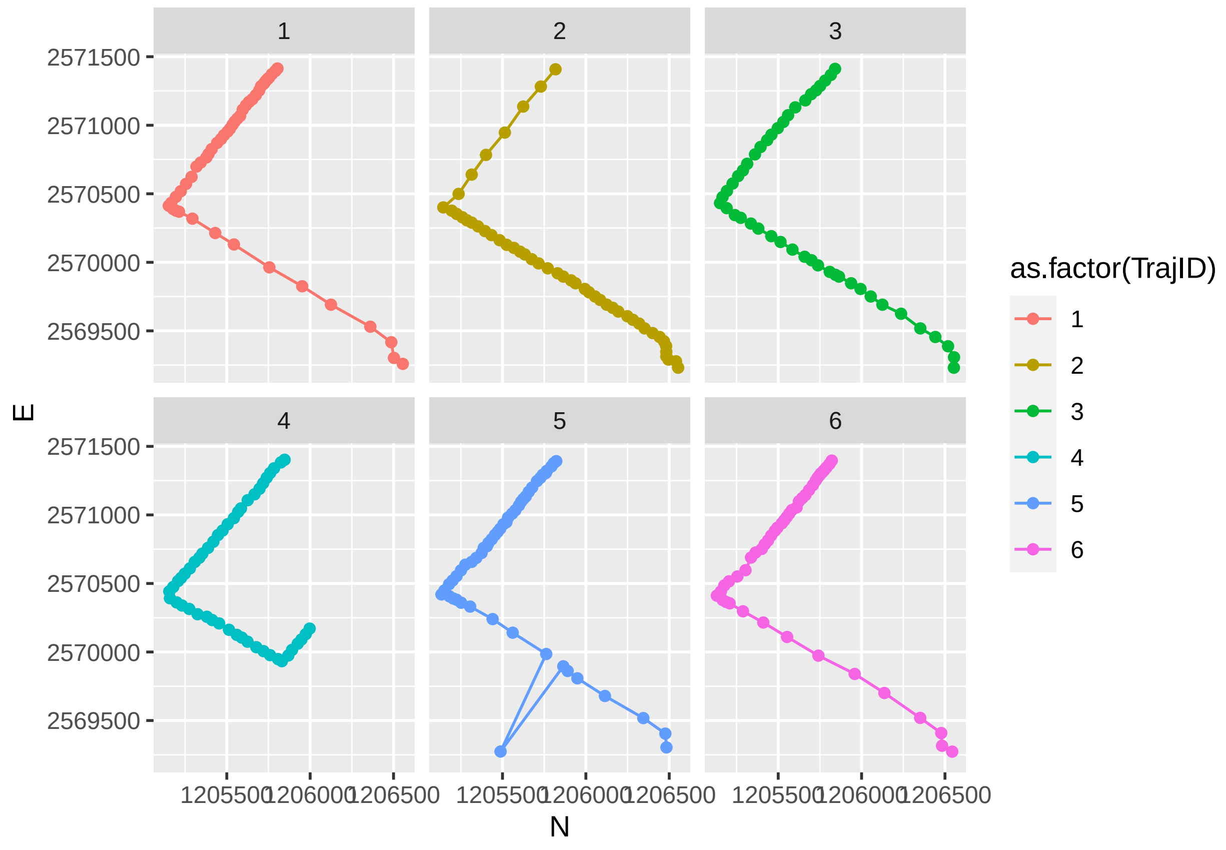
Importing dataset

```
pedestrians <- read_csv("pedestrian.csv")
```

```
## Rows: 289 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (3): TrajID, E, N
## dttm (1): DatetimeUTC
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Visual comparision of the 6 trajectories

```
ggplot(pedestrians, aes(N,E, col = as.factor(TrajID)))+
  geom_point() +
  geom_path() +
  facet_wrap(~TrajID)
```



Task 6: Calculate similarity

First I create the matrices

```
pedestrians1 <- pedestrians %>%
  filter(TrajID == 1) %>%
  select(E,N)%>%
  as.matrix()

pedestrians2 <- pedestrians %>%
  filter(TrajID == 2) %>%
  select(E,N) %>%
  as.matrix()

pedestrians3 <- pedestrians %>%
  filter(TrajID == 3) %>%
  select(E,N) %>%
  as.matrix()

pedestrians4 <- pedestrians %>%
  filter(TrajID == 4) %>%
  select(E,N) %>%
  as.matrix()

pedestrians5 <- pedestrians %>%
```

```r
  filter(TrajID == 5) %>%
  select(E,N) %>%
  as.matrix()

pedestrians6 <- pedestrians %>%
  filter(TrajID == 6) %>%
  select(E,N) %>%
  as.matrix()
```

Now I compare trajectory 1 to trajectories 2-6 using different similarity measures from the package SimilarityMeasures

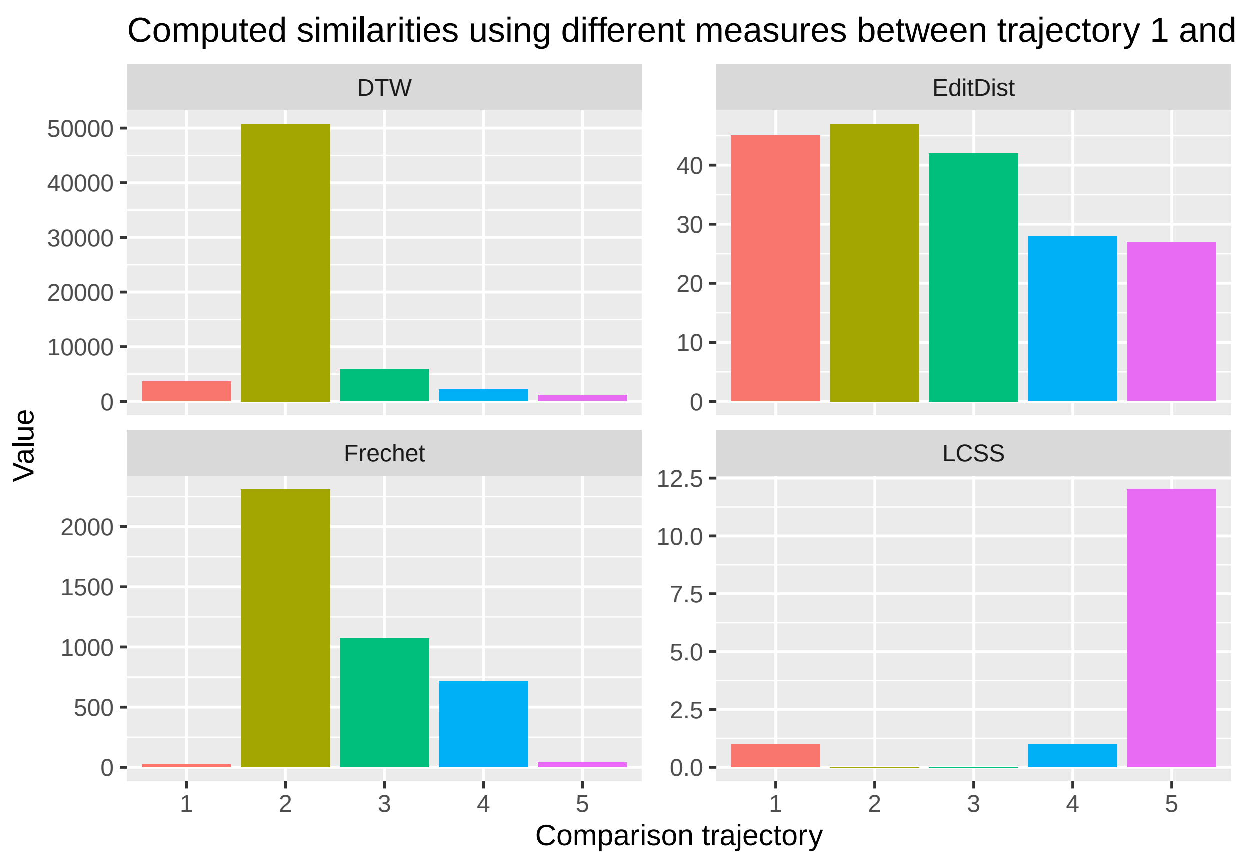
```r
DTW = c(DTW(pedestrians1, pedestrians2),
        DTW(pedestrians1, pedestrians3),
        DTW(pedestrians1, pedestrians4),
        DTW(pedestrians1, pedestrians5),
        DTW(pedestrians1, pedestrians6))

EditDist = c(EditDist(pedestrians1, pedestrians2),
             EditDist(pedestrians1, pedestrians3),
             EditDist(pedestrians1, pedestrians4),
             EditDist(pedestrians1, pedestrians5),
             EditDist(pedestrians1, pedestrians6))

Frechet = c(Frechet(pedestrians1, pedestrians2),
            Frechet(pedestrians1, pedestrians3),
            Frechet(pedestrians1, pedestrians4),
            Frechet(pedestrians1, pedestrians5),
            Frechet(pedestrians1, pedestrians6))

LCSS = c(LCSS(pedestrians1, pedestrians2, pointDistance = 10, pointSpacing = 0, errorMarg = 1),
         LCSS(pedestrians1, pedestrians3, pointDistance = 10, pointSpacing = 0, errorMarg = 1),
         LCSS(pedestrians1, pedestrians4, pointDistance = 10, pointSpacing = 0, errorMarg = 1),
         LCSS(pedestrians1, pedestrians5, pointDistance = 10, pointSpacing = 0, errorMarg = 1),
         LCSS(pedestrians1, pedestrians6, pointDistance = 10, pointSpacing = 0, errorMarg = 1))
```

Now I create a new data frame containing the results

```r
similarity_DF <- data.frame(ID = seq(1:5), DTW, EditDist, Frechet, LCSS)
```

Plotting the Results, I first transform the data frame from wide to long format and then I plot the results

```r
similarity_DF <- similarity_DF %>%
  pivot_longer(-ID, names_to = "Measures", values_to = "Value")

ggplot(similarity_DF, aes(as.factor(ID), Value, fill = as.factor(ID))) +
  geom_bar(stat = "identity") +
  facet_wrap(~Measures, scales = "free_y") +
  theme(legend.position = "none") +
  labs(title = "Computed similarities using different measures between trajectory 1 and all the others"
       y = "Value", x = "Comparison trajectory")
```



Computed similarities using different measures between trajectory 1 and
DTW
EditDist
Frechet
LCSS
Value
Comparison trajectory
50000
40000
30000
20000
10000
0
40
30
20
10
0
2000
1500
1000
500
0
12.5
10.0
7.5
5.0
2.5
0.0
1
2
3
4
5
1
2
3
4
5