CMA - Exercise 5

kinmar01

Table of contents

exercise 5, part of the course cma (mainly based on Laube (2014))

Setup

```
pacman::p_load("readr", "sf", "dplyr", "ggplot2","RColorBrewer","tidyr","lubridate","knitr")

theme_minimal() |> theme_set()
```

1 Write your own functions

Create the following two functions:

1. A function which calculates a persons BMI based on their height and weight (Equation 1)

$$BMI = \frac{\text{Weight (kg)}}{\text{Height (m)}^2} \tag{1}$$

```r
bmi <- function(weight,height) {
  weight/(height)^2
}
```

2. A function which converts degrees Celcius to Farenheight (Equation 2)

$$Farenheight = Celsius * 1.8 + 32 \tag{2}$$

```r
farenheight <- function(celsius) {
  celsius * 1.8 + 32
}
```

3. A function which calculates the (Euclidean) distance between two sets of coordinates (x1, y1 and x2, y2) (Equation 3)

$$\text{Euclidean Distance} = \sqrt{(x_2 - x_1)^2 + (x_2 - y_1)^2} \tag{3}$$

```r
euclidean_distance <- function(x1, y1, x2, y2) {
  sqrt((x2 - x1)^2 + (y2 - y1)^2)
}
```

2 Prepare Analysis

In the next tasks we will look for "meet" patterns in our wild boar data. To simplify this, we will only use a subset of our wild boar data: The individuals *Rosa* and *Sabi* for the timespan *01.04.2015 - 15.04.2015*. Use the dataset `wildschwein_BE_2056.csv` (on moodle). Import the csv as a data.frame and filter it with the aforementioned criteria. You do not need to convert the data.frame to an `sf` object.

```r
df_wild_boar <- read_delim("data/wildschwein_BE_2056.csv") |>
  mutate(across(where(is.character), as.factor)) |>
  filter(
    TierName %in% c("Rosa", "Sabi"),
    DatetimeUTC |> between(
      as.POSIXct("2015-04-01 00:00:00", tz = "UTC"),
      as.POSIXct("2015-04-15 23:59:59", tz = "UTC"))
  )
```

```
Rows: 51246 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.
```

```
df_wild_boar |> str()
```

```
tibble [2,860 x 6] (S3: tbl_df/tbl/data.frame)
 $ TierID     : Factor w/ 3 levels "002A","016A",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ TierName   : Factor w/ 3 levels "Rosa","Ruth",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ CollarID   : num [1:2860] 12275 12275 12275 12275 12275 ...
 $ DatetimeUTC: POSIXct[1:2860], format: "2015-04-01 00:00:11" "2015-04-01 00:15:22" ...
 $ E          : num [1:2860] 2570372 2570309 2570326 2570315 2570323 ...
 $ N          : num [1:2860] 1205313 1205262 1205248 1205242 1205237 ...
```

```
df_wild_boar |> summary()
```

```
  TierID     TierName       CollarID      DatetimeUTC                  
 002A:1440   Rosa:1420   Min.   :12275   Min.   :2015-04-01 00:00:10.00  
 016A:1420   Ruth:   0   1st Qu.:12275   1st Qu.:2015-04-04 17:15:12.75  
 018A:   0   Sabi:1440   Median :12275   Median :2015-04-08 10:37:41.00  
                         Mean   :13118   Mean   :2015-04-08 10:40:50.61  
                         3rd Qu.:13972   3rd Qu.:2015-04-12 04:00:17.50  
                         Max.   :13972   Max.   :2015-04-15 23:47:56.00  
       E                N          
 Min.   :2569715   Min.   :1202620  
 1st Qu.:2569784   1st Qu.:1204908  
 Median :2570347   Median :1205182  
 Mean   :2570558   Mean   :1204930  
 3rd Qu.:2570638   3rd Qu.:1205207  
 Max.   :2574355   Max.   :1205669  
```

```
df_wild_boar |> head() |> kable()
```

Table 1: Wild boar data

TierID	TierName	CollarID	DatetimeUTC	E	N
002A	Sabi	12275	2015-04-01 00:00:11	2570372	1205313
002A	Sabi	12275	2015-04-01 00:15:22	2570309	1205262
002A	Sabi	12275	2015-04-01 00:30:11	2570326	1205248
002A	Sabi	12275	2015-04-01 00:45:16	2570315	1205242
002A	Sabi	12275	2015-04-01 01:00:44	2570323	1205237
002A	Sabi	12275	2015-04-01 01:15:17	2570320	1205247

3 Create Join Key

Have a look at your dataset. You will notice that samples are taken at every full hour, quarter past, half past and quarter to. The sampling time is usually off by a couple of seconds.

To compare Rosa and Sabi's locations, we first need to match the two animals *temporally*. For that we can use a `join`, but need *identical* time stamps to serve as a join key. We therefore need to slightly adjust our time stamps to a common, concurrent interval.

The task is therfore to round the minutes of `DatetimeUTC` to a multiple of 15 (00, 15, 30, 45) and store the values in a new column.

```
df_wild_boar_t3 <- df_wild_boar |>
  mutate(
    DatetimeRound = DatetimeUTC |> round_date(unit = "15 minutes")
  )
```

```
df_wild_boar_t3 |> head() |> kable()
```

Table 2: Wild boar data with rounded datetime.

TierID	TierName	CollarID	DatetimeUTC	E	N	DatetimeRound
002A	Sabi	12275	2015-04-01 00:00:11	2570372	1205313	2015-04-01 00:00:00
002A	Sabi	12275	2015-04-01 00:15:22	2570309	1205262	2015-04-01 00:15:00
002A	Sabi	12275	2015-04-01 00:30:11	2570326	1205248	2015-04-01 00:30:00
002A	Sabi	12275	2015-04-01 00:45:16	2570315	1205242	2015-04-01 00:45:00

Table 2: Wild boar data with rounded datetime.

TierID	TierName	CollarID	DatetimeUTC	E	N	DatetimeRound
002A	Sabi	12275	2015-04-01 01:00:44	2570323	1205237	2015-04-01 01:00:00
002A	Sabi	12275	2015-04-01 01:15:17	2570320	1205247	2015-04-01 01:15:00

4 Measuring distance at concurrent locations

To measure the distance between concurrent locations, we need to follow the following steps.

1. Split the `df_wild_boar_t3` object into one `data.frame` per animal

```
df_wild_boar_Rosa <- df_wild_boar_t3 |>
  filter(TierName == "Rosa")

df_wild_boar_Sabi <- df_wild_boar_t3 |>
  filter(TierName == "Sabi")
```

2. Join these datasets by the new `Datetime` column created in the last task. The joined observations are *temporally close*.

```
df_wild_boar_joined <- left_join(
  df_wild_boar_Sabi,
  df_wild_boar_Rosa,
  join_by(DatetimeRound),
  suffix = c("_Sabi", "_Rosa")
)
```

3. In the joined dataset, calculate Euclidean distances between concurrent observations and store the values in a new column

```
df_wild_boar_dist <- df_wild_boar_joined |>
  mutate(
    dist = euclidean_distance(E_Sabi,N_Sabi,E_Rosa,N_Rosa)
  )
```

4. Use a reasonable threshold on `distance` to determine if the animals are also *spatially close* enough to constitute a *meet* (we use 100 meters). Store this Boolean information (`TRUE`/`FALSE`) in a new column

```
df_wild_boar_t4 <- df_wild_boar_dist |>
  mutate(
    meet = (dist^2)^0.5 <= 100
  )
```

5 Visualize data

Now, visualize the *meets* spatially in a way that you think reasonable. For example in the plot as shows below. To produce this plot we:

- Used the individual dataframes from `rosa` and `sabi` (from the previous task)
- Used the joined dataset (also from the previous task), filtered to only the meets
- Manually changed the x and y axis limits

```
df_wild_boar_t4 |>
  filter(meet) |>
  arrange(DatetimeRound) |>
  ggplot()+
    geom_point(data=df_wild_boar_Rosa,aes(E,N,color=TierName),alpha=0.2)+
    geom_point(data=df_wild_boar_Sabi,aes(E,N,color=TierName),alpha=0.2)+
    geom_point(aes(E_Rosa,N_Rosa,fill=TierName_Rosa), shape = 21, size = 2,color="black")+
    geom_point(aes(E_Sabi,N_Sabi,fill=TierName_Sabi), shape = 21, size = 2,color="black")+
  coord_equal()+
  guides(
    color = guide_legend(title = "Regular Locations"),
    fill = guide_legend(title = "Meets")
  )
```

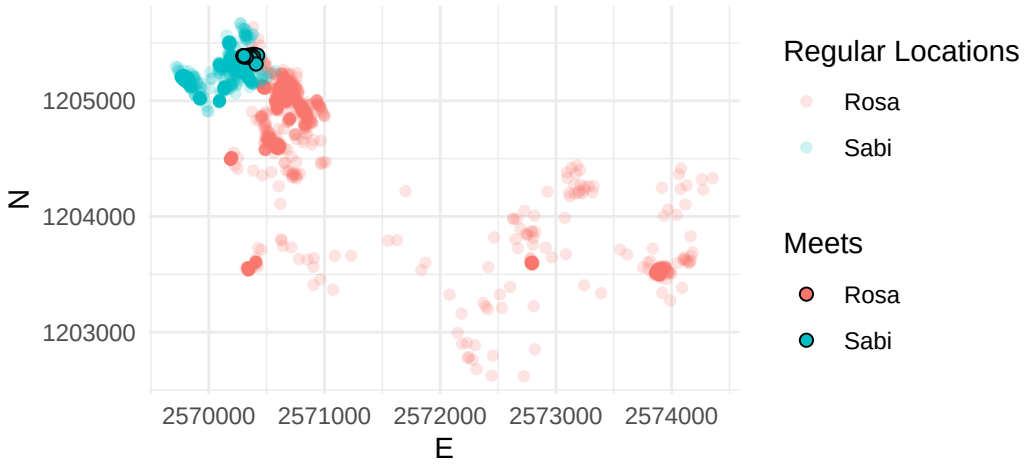

Figure 1: Wild boars; where Rosa and Sabi meets

6 References

Laube, Patrick. 2014. *Computational Movement Analysis*. 2014th ed. SpringerBriefs in Computer Science. Cham: Springer International Publishing AG.