Data extraction tutorial for experimental human approach trials on wild, GPS-collared wolves

Supplementary Presentation 3

Eriksen A, Versluijs E, Fuchs B, Zimmermann B, Wabakken P, Ordiz A, Sunde P, Wikenros C, Sand H, Gillich B, Michler F, Nordli K, Carricondo-Sanchez D, Gorini L and Rieger S (2022). A Standardized Method for Experimental Human Approach Trials on Wild Wolves. Front. Ecol. Evol. 10:793307. doi: 10.3389/fevo.2022.793307

Contents

1 Introduction

This tutorial describes how GPS data from experimental human approaches on wolves can be prepared and analysed as described in *Eriksen & Versluijs, et.al. (2022)*. To illustrate this method, we used two data-sets from an approach trial in which two observers approached a wolf on **30.01.2020** following the field protocol described in *Eriksen & Versluijs, et.al. (2022) Supplementary Presentation 1*. The data-sets includes GPS data from the wolf's collar (Vectronics VERTEX Plus), and the track-log from a hand held GPS unit (Garmin gpsmap 62) carried by the observers.

The following steps are described:

- Data preparation: prepare the data for flight initiation and resettling analysis
- Detecting the flight initiation with changepoint analysis
- Detecting the wolf's resettling position with changepoint analysis

- Extracting variables based on the moment of flight initiation and resettling (flight duration, total distance traveled, displacement, initial and overall speed, and initial and overall straightness)

2 Setup

The packages needed to run the code:

```
library(readr)         # Reading the CSV file
library(dplyr)         # Data manipulation
library(lubridate)     # Handling dates and time formats
library(plotKML)       # Reading GPX files
library(sp)            # Re-projecting GPS coordinates
library(ggplot2)       # Visualizing data
library(changepoint)   # Running changepoint analysis
library(trajr)         # Package for animal trajectory analysis
library(flextable)     # Package for making tables
```

3 Data preparation

In this step we extract the wolf's GPS data which only includes the *approach* and *post-disturbance* periods. Additionally, we also extract the observer data and combine it with the wolf GPS data. It is advised to visualize the approach trial in e.g. Qgis or within the software R, for example, using the package *Movevis*. The example wolf data is in CSV format, while the observer data is in GPX (which includes tracks, way-points and routes).

3.1 Load and prepare data

Firstly, we load the data into the environment, using the *'reader'* package. In our case we have a *semicolon* as delimiter (delim = ";"). However, in many occasions this might be a *comma*.

```
wolves <- read_delim(
  file= "wolf_data.csv",
  delim = ";", col_names = TRUE)
```

After loading the CSV, we can extract the relevant information for the defined period. In this example we subset the relevant columns from the wolf GPS data and transform the *DataTime* column to a POSTIxct time format and filter for the period between 12:00:00 and 17:05:00. This period includes both the *approach* and *post-disturbance* periods. To ensure that the last position of the post-trial period is captured in our data frame, we added extra five minutes for the selection. Additionally, we add an approachID (in this case we call it *'Example approach'*) to make it easier to link extracted variables back to this approach trial.

```
wolves <- wolves %>%
  select(DateTime,
         wolfID = IndividId,
         Latitude,
         Longitude,
         UTMZone,
         UTMX,
         UTMY) %>%
  mutate(DateTime = dmy_hms(DateTime, tz ="Etc/GMT-1")) %>%
  filter(DateTime >= "2020-01-30 12:00:00" & DateTime <= "2020-01-30 17:05:00") %>%
  mutate(approachID = "Example approach")
```

Depending on how the data from the wolf collars is obtained (e.g. directly from the collar or downloaded remotely), there is a possibility that the data from two wolves during one approach trial is combined into one

csv. Therefore, the data needs to be divided into different data frames. In our example, only one wolf is present. However, here we show how you can extract a single wolf based on the collar id-number.

```
wolf <- wolves %>%
  filter(wolfID == "M18-13")
```

When the wolf data is prepared a brief inspection of the first rows is useful to intercept any mistakes or errors.

```
head(wolf)
```

```
## # A tibble: 6 x 8
##   DateTime            wolfID Latitude Longitude UTMZone   UTMX   UTMY approachID
##   <dttm>              <chr>     <dbl>     <dbl>   <dbl>  <dbl>  <dbl> <chr>
## 1 2020-01-30 12:00:10 M18-13     60.9      12.5      33 363926 6.75e6 Example a~
## 2 2020-01-30 12:01:08 M18-13     60.9      12.5      33 363923 6.75e6 Example a~
## 3 2020-01-30 12:02:06 M18-13     60.9      12.5      33 363925 6.75e6 Example a~
## 4 2020-01-30 12:03:08 M18-13     60.9      12.5      33 363923 6.75e6 Example a~
## 5 2020-01-30 12:04:08 M18-13     60.9      12.5      33 363925 6.75e6 Example a~
## 6 2020-01-30 12:05:08 M18-13     60.9      12.5      33 363923 6.75e6 Example a~
```

The observer data comes in a GPX format, therefore we use the function 'readGPX' from the *'plotKML'* package to load the data. Thereafter we extract the saved positions during the approach trial to a new data frame. In most cases the data is saved under 'tracks', in our case it was saved under 'waypoints'.

```
observer <- readGPX("observer_tracklog.gpx")

observer <- data.frame(observer$waypoints)
names(observer)
```

```
## [1] "lon"  "lat"  "time"
```

Often we want to change the column names from the original GPX file to aid readability. Often the column names are long and unreadable, e.g. 'X2020.01.30.13.16.32.Dag.lon'. The column 'time' is renamed to 'DateTime' to match the similar column in the wolf data. The data is filtered for the duration of the 1-minute period. Filtering on shorter periods is possible as long as the data includes the whole trial period.

```
observer <- observer %>%
  select(lon = lon,
         lat = lat,
         DateTime = time) %>%
  mutate(DateTime = ymd_hms(DateTime, tz = "Etc/GMT-1")) %>%
  filter(DateTime >= ymd_hms("2020-01-30 12:00:00", tz = "Etc/GMT-1") &
         DateTime <= ymd_hms("2020-01-30 14:30:00", tz = "Etc/GMT-1"))
```

By looking at the first rows we can determine any errors. In this example we see that the GPS track starts at 12:56 and recorded with 1-second intervals.

```
head(observer)
```

```
##        lon      lat            DateTime
## 1 12.48137 60.91016 2020-01-30 12:56:19
## 2 12.48137 60.91016 2020-01-30 12:56:20
## 3 12.48138 60.91016 2020-01-30 12:56:21
## 4 12.48138 60.91016 2020-01-30 12:56:22
## 5 12.48138 60.91016 2020-01-30 12:56:23
## 6 12.48138 60.91016 2020-01-30 12:56:24
```

The observer data is recorded in the geographic coordinate system *'WGS84 (EPSG:4326)'*. We need to transform coordinates into a projected coordinate system, for our data this would be *'WGS84 UTM zone*

33N (EPSG:32633)'. We use the function 'spTransform' from the *'sp'* package to add the UTM coordinates. By viewing the first five rows we see that the obsX and obsY contains now UTM coordinates.

```
observer$obsX <- observer$lon
observer$obsY <- observer$lat

coordinates(observer) <- c("obsX", "obsY")
proj4string(observer) <- CRS("EPSG:4326")

observer <- spTransform(observer, CRS("EPSG:32633"))

observer <- as.data.frame(observer)
head(observer)
```

```
##        lon      lat            DateTime     obsX    obsY
## 1 12.48137 60.91016 2020-01-30 12:56:19 363416.6 6755405
## 2 12.48137 60.91016 2020-01-30 12:56:20 363416.7 6755405
## 3 12.48138 60.91016 2020-01-30 12:56:21 363416.8 6755405
## 4 12.48138 60.91016 2020-01-30 12:56:22 363416.9 6755405
## 5 12.48138 60.91016 2020-01-30 12:56:23 363417.0 6755405
## 6 12.48138 60.91016 2020-01-30 12:56:24 363417.1 6755405
```

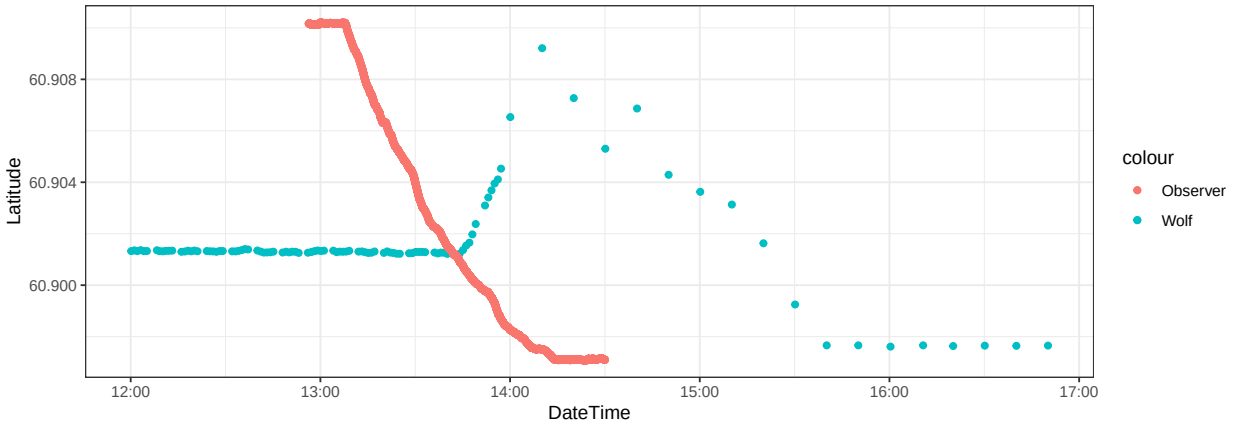
We combine the wolf data with the observer data and calculate the distance (in meters) between the observer and the wolf during the approach trial. The hand-held GPS recorded in 1-second interval, which gives the possibility to join the data frames based on 'DateTime'. We use the 'left_join' function from the *'Dplyr'* package to join the 'observer' data with the 'wolf' data frame. When the join is successful we added a new column called 'dist_wolf_obs' and calculated the Euclidean distance between the observer and wolf.

```
wolf <- left_join(wolf, observer, by = "DateTime")

wolf$dist_wolf_obs <- NA

for(i in 1:nrow(wolf)){
  wolf$dist_wolf_obs[i] <- sqrt((wolf$UTMX[i]-wolf$obsX[i])^2 +
                                  (wolf$UTMY[i]-wolf$obsY[i])^2)
}

summary(wolf$dist_wolf_obs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   89.89  299.84  610.04  683.92 1074.28 2155.88      58
```

3.2 Visualize data

Visualizing the data by plotting 'DateTime' against 'latitude' (or 'longitude') gives a first impression of the approach trial and the quality of the data. The following graph shows the DateTime against latitude and gives an indication if data points are missing (gaps between consecutive data positions). It also shows if the wolves responded spatially to the observer. With no movement, the latitude (or longitude) stays in a horizontal line.



4 Flight initiation distance (FID)

4.1 Prepare data for analysis

For detecting the flight initiation distance (FID) based on Changepoint analysis, we need to extract the 1-minute positioning intervals. Since we need to data from the original prepared data later on, we prepare the data into a new data frame (*wolf_fid*).

```
wolf_fid <- wolf %>%
  filter(DateTime >= ymd_hms("2020-01-30 12:00:00", tz = "Etc/GMT-1") &
         DateTime <= ymd_hms("2020-01-30 14:05:00", tz = "Etc/GMT-1"))
```

Hereafter, we can calculate the difference in time between consecutive positions using the function 'difftime', and we calculate the euclidean distance and the speed in meters per minute (m/min)between consecutive positions.

```
wolf_fid$difftime <- c(NA, difftime(wolf_fid$DateTime[-1],
                     wolf_fid$DateTime[-nrow(wolf_fid)],
                     units = "secs")
)

wolf_fid$dist <- NA
for(i in 2:nrow(wolf_fid)){
  wolf_fid$dist[i] <- sqrt((wolf_fid$UTMX[i]-wolf_fid$UTMX[i-1])^2 +
                           (wolf_fid$UTMY[i]-wolf_fid$UTMY[i-1])^2)
}

wolf_fid$speed <- (wolf_fid$dist/wolf_fid$difftime)*60

wolf_fid <- wolf_fid[-1,]

summary(wolf_fid$speed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.876   3.162   8.582   5.385  97.755
```

We then adjust the data to a gamma distribution by changing speed to 0.01 m/min when this was 0 m/min.

```
wolf_fid <- wolf_fid %>%
  mutate(speed = if_else(speed == 0, true = 0.01, false = speed))
```

4.2 Changepoint analysis

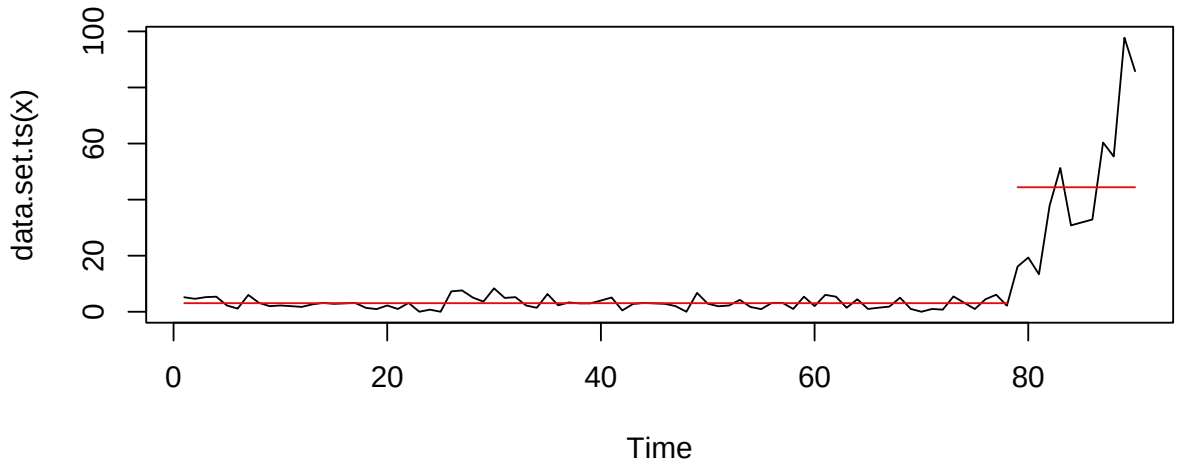
We run changepoint analysis with the function *cpt.meanvar* from the *'changepoint'* package, using the pruned exact linear time (PELT) algorithm and a Gamma distribution. The penalty is set to *"MBIC"*. However, in cases where changepoint analysis did not find any changepoints, but visual inspection showed a clearly a spatial response of the wolf towards the observer the penalty can be set to *"AIC"*.

```
wolf_cpt <- cpt.meanvar(wolf_fid$speed,
                                method = "PELT",
                                penalty = "MBIC",
                                test.stat = "Gamma")
```

4.3 Detecting flight initiation

The moment of flight initiation is defined as the wolf's position at the first changepoint after the trial has started. In this case the approach trial started at 13:07, therefore the changepoint must be after this moment.

```
plot(wolf_cpt)
```




```
# DateTime from the changepoints
wolf_fid[cpts(wolf_cpt),]
```

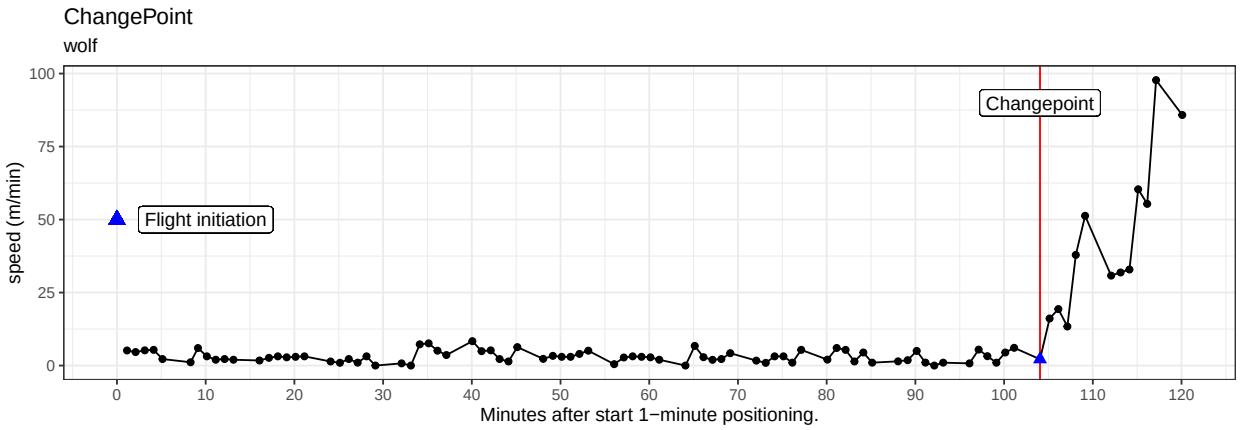
```
## # A tibble: 1 x 16
##   DateTime            wolfID Latitude Longitude UTMZone   UTMX   UTMY approachID
##   <dttm>              <chr>     <dbl>     <dbl>   <dbl>  <dbl>  <dbl> <chr>
## 1 2020-01-30 13:44:03 M18-13     60.9      12.5      33 363969 6.75e6 Example a~
## # ... with 8 more variables: lon <dbl>, lat <dbl>, obsX <dbl>, obsY <dbl>,
## #   dist_wolf_obs <dbl>, difftime <dbl>, dist <dbl>, speed <dbl>
```

In our case the changepoint is after the time when the approach started. The flight initiation can be extracted from the 'wolf_fid' data frame and the flight initiation distance can be found in the column 'dist_wolf_obs'.

```
wolf_changepoint <- wolf_fid[cpts(wolf_cpt),]

wolf_changepoint$dist_wolf_obs
```

```
## [1] 89.89095
```

The flight initiation can be shown graphically as:



5 Wolf resettling position

We can calculate the wolf's resettling position using changepoint analysis, which includes the following steps:

- Down-sample the 1-minute intervals to 10-minute intervals
- Calculate distance and speed based on 10-minute positions
- Run *cpt.meanvar* changepoint analysis

5.1 Prepare data for analysis

The data frame 'wolf' consist both 1-minute and 10-minute interval data, therefore we first need to down-sample the 1-minute interval data to 10-minute intervals. We do this by creating a new data frame consisting of exactly 10-minute intervals for the duration of the trial and post-trial period.

```
resettling_10min <- seq.POSIXt(as.POSIXct("2020-01-30 12:00", tz = "Etc/GMT-1"),
                               as.POSIXct("2020-01-30 17:00", tz = "Etc/GMT-1"),
                       by = "10 min")
```

Now we can join the data frame 'wolf' with the 'resettling_10min' by using the *'data.table'* package. We transform the data frames into the 'data.table' format and define the column 'DateTime' in 'wolf_res' as the key link between the two data frames. If this is set, we can extract the data from the wolf_res to the 'resettling_10min' by using *'roll = "nearest"'*. This function from *'data.table'* extract the 'nearest' observation of 'DateTime from the 'wolf_res based on the 'resettling_10min' data frame. We can check the first five rows to check if the join is successful. Note that we call directly to the *'data.table'* package, instead of loading it via the function 'library()', this is to avoid problems as *'data.table'* masks some functions from the *'dplyr'* package.

```
resettling_10min <- data.table::data.table(resettling_10min)
wolf_res <- data.table::data.table(wolf)

data.table::setkey(wolf_res, DateTime)

wolf_res <- wolf_res[resettling_10min, roll = "nearest"]

head(wolf_res)
```

```
##              DateTime wolfID Latitude Longitude UTMZone   UTMX    UTMY
## 1: 2020-01-30 12:00:00 M18-13 60.90132  12.49148      33 363926 6754399
## 2: 2020-01-30 12:10:00 M18-13 60.90132  12.49145      33 363925 6754398
```

```
## 3: 2020-01-30 12:20:00 M18-13 60.90135  12.49142      33 363923 6754402
## 4: 2020-01-30 12:30:00 M18-13 60.90133  12.49142      33 363923 6754400
## 5: 2020-01-30 12:40:00 M18-13 60.90135  12.49217      33 363964 6754401
## 6: 2020-01-30 12:50:00 M18-13 60.90128  12.49232      33 363972 6754392
##          approachID lon lat obsX obsY dist_wolf_obs
## 1: Example approach  NA  NA   NA   NA            NA
## 2: Example approach  NA  NA   NA   NA            NA
## 3: Example approach  NA  NA   NA   NA            NA
## 4: Example approach  NA  NA   NA   NA            NA
## 5: Example approach  NA  NA   NA   NA            NA
## 6: Example approach  NA  NA   NA   NA            NA
```

With the down-sampled data we can calculate the distance and speed between the consecutive positions. The time difference is known in this case, as we sampled to exact 10-minute intervals.

```
wolf_res$dist <- NA
for(i in 2:nrow(wolf_res)){
  wolf_res$dist[i] <- sqrt((wolf_res$UTMX[i]-wolf_res$UTMX[i-1])^2 +
                           (wolf_res$UTMY[i]-wolf_res$UTMY[i-1])^2)
}

wolf_res$speed <- (wolf_res$dist/10)

wolf_res <- wolf_res[-1,]
wolf_res$id <- seq(1, nrow(wolf_res))
```

In this last step we change the speeds from 0 to 0.01 metres per minute to ensure a 'Gamma' distribution.

```
wolf_res <- wolf_res %>%
  mutate(speed = if_else(speed == 0, true = 0.01, false = speed))
```

5.2 Changepoint analysis

The changepoint analysis are similar as for the flight initiation.

```
wolf_res_cpt <- cpt.meanvar(wolf_res$speed,
                                  method = "PELT",
                                  penalty = "MBIC",
                                  test.stat = "Gamma")
```

5.3 Detecting the resettling position

Resettling is initiated during the first changepoint after the flight initiation, therefore the position after this changepoint is defined as the resettling position. First we extract all the changepoints to a new data frame, then we can define which changepoint was the first after the flight initiation. Thereafter, we can extract the data for the position after this changepoint and extract the data to a new data frame 'wolf_resettling'.

```
wolf_res_cp <- wolf_res[cpts(wolf_res_cpt),]

wolf_res_cp
```

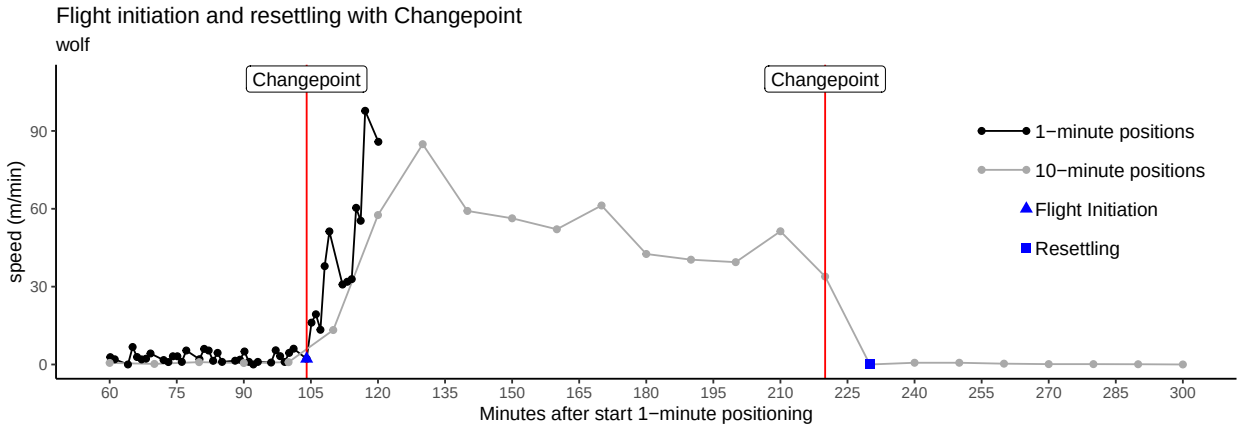
```
##               DateTime wolfID Latitude Longitude UTMZone   UTMX    UTMY
## 1: 2020-01-30 13:40:00 M18-13 60.90122  12.49233      33 363972 6754386
## 2: 2020-01-30 15:40:00 M18-13 60.89766  12.39692      33 358783 6754192
##          approachID      lon      lat     obsX    obsY dist_wolf_obs       dist
## 1: Example approach 12.48978 60.90152 363835.3 6754425      142.3224   8.944272
## 2: Example approach       NA       NA       NA      NA            NA 338.881985
```

```
##         speed id
## 1:  0.8944272 10
## 2: 33.8881985 22
```

```
wolf_resettling <- wolf_res %>%
  filter(id == (wolf_res_cp[2,]$id + 1))

wolf_resettling
```

```
##               DateTime wolfID Latitude Longitude UTMZone   UTMX    UTMY
## 1: 2020-01-30 15:50:00 M18-13 60.89766  12.39692      33 358783 6754192
##          approachID lon lat obsX obsY dist_wolf_obs dist speed id
## 1: Example approach  NA  NA   NA   NA            NA    0  0.01 23
```

The resettling can be shown graphically as:



6 Extracting flight variables

Other variables which can be extracted are based on the flight initiation and resettling position.

6.1 Calculating flight variables

First we create a new data frame where we select the approach and wolf details (approachID and wolfID), which also includes the flight initiation distance and the closest distance between observer and wolf.

```
wolf_data_summary <- wolf_changepoint %>%
  select(approachID = approachID,
         wolfID = wolfID,
         fid = dist_wolf_obs) %>%
  mutate(min_obs_wolf = min(na.omit(wolf$dist_wolf_obs)))
```

We add the time from the observer passing the passing position, obtained from the field form, in the same format as the 'DateTime' column. Now we can calculate the time difference with the flight initiation using the *'difftime'* function.

```
pp_flight_timediff <- ymd_hms("2020-01-30 13:42:00", tz = "Etc/GMT-1")
wolf_data_summary$pp_flight_diff <- difftime(pp_flight_timediff,
                                             wolf_changepoint$DateTime,
                                             units = "mins")
```

The same function (*'difftime'*) can be applied to calculate the total duration of the flight by calculating the difference in minutes between the flight initiation and resettling.

```
wolf_data_summary$flight_duration <- difftime(wolf_resettling$DateTime,
                                              wolf_changepoint$DateTime,
                                              units = "mins")
```

The initial flight speed, i.e. the average speed for the first 10 minutes of the flight, is calculated by extracting only the first 10 minutes of the flight from the 1-minute positioning interval data frame (*wolf_fid*). Hereafter the average speed can be calculated.

For the overall speed, i.e. the average speed for the whole flight, the method is the same. We use the 10-minute positioning interval data frame to extract the total flight and to calculate the average speed.

```
wolf_data_summary$initial_speed <- wolf_fid %>%
  filter(DateTime >= wolf_changepoint$DateTime &
         DateTime <= wolf_changepoint$DateTime + minutes(10)) %>%
  summarise(initial_speed = mean(speed)) %>%
  pull(initial_speed)

wolf_data_summary$overall_speed  <- wolf_res %>%
  filter(DateTime >= wolf_changepoint$DateTime &
         DateTime <= wolf_resettling$DateTime) %>%
  summarise(overall_speed = mean(speed)) %>%
  pull(overall_speed)
```

To obtain the distance, displacement and overall straightness we used the *'trajr'* package. Where we first change the data frame to a trajr object and then we use the function 'TrajLength' to calculate the total distance traveled and the function 'TrajDistance' to calculate the linear displacement of the wolf.

```
wolf_10min_trajr <- wolf_res %>%
  filter(DateTime >= wolf_changepoint$DateTime &
         DateTime <= wolf_resettling$DateTime) %>%
  select(x = UTMX,
         y = UTMY,
         time = DateTime)

wolf_10min_trajr <- TrajFromCoords(wolf_10min_trajr)

wolf_data_summary$flight_distance <- TrajLength(wolf_10min_trajr)
wolf_data_summary$flight_displacement <- TrajDistance(wolf_10min_trajr)
```

The total flight duration is not on the same between different approaches. Therefore, we adjust the overall straightness by calculating the average straightness index across the flight based on the straightness between every three consecutive positions.

```
    for(i in 1:nrow(wolf_10min_trajr)) {
      wolf_10min_trajr$straightness[i] <- TrajStraightness(wolf_10min_trajr[i:(i+2),])
    }

wolf_data_summary$overall_straightness <- mean(na.omit(wolf_10min_trajr$straightness))
```

For the initial straightness we also use the *'trajr'* package, but then for only the first 10 minutes of the flight. We can now also calculate the initial straightness with the 'TrajStraightness' function.

```
wolf_1min_trajr <- wolf_fid%>%
  filter(DateTime >= wolf_changepoint$DateTime &
         DateTime <= wolf_changepoint$DateTime + minutes(10)) %>%
  select(x = UTMX,
         y = UTMY,
```

```
         time = DateTime)

wolf_1min_trajr <- TrajFromCoords(wolf_1min_trajr)

wolf_data_summary$initial_straightness <- TrajStraightness(wolf_1min_trajr)
```

6.2 Final approach variables

Now we can look at all the variables we obtained based on the flight initiation and resettling of the wolf.

Variable name	Result
ApproachID	Example approach
WolfID	M18-13
Flight/No flight	Flight
Minimum wolf-observer distance	89.9
Flight initiation distance (FID)	89.9
Passing-flight time difference	-2.05 mins
Initial speed	25.4
Initial straightness	0.96
Flight duration	126 mins
Flight displacement	5172
Total distance traveled	5790
Overall speed	45.6
Overall straightness	0.975