Plot positions only

Plots the GPS positions as eastings and northings on a map.

TODO: When printing the segment and jump speeds, need to also print the dates that go with the jumps, make a joint tbale for all units

```
rm(list=ls())
setwd("~/WORKSHOP/GPS/")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(dsm)
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: mrds
```

```
## This is mrds 2.2.6
## Built: R 3.6.3; ; 2022-06-17 10:42:29 UTC; unix
```

```
## Loading required package: numDeriv
```

```
## This is dsm 2.3.2
## Built: R 3.6.3; ; 2022-06-17 10:42:38 UTC; unix
```

```
library(anytime)

Rearth <- 6371*1e3 # meters

data <- readRDS("OUTPUT/complete_GPS_data.rds")
data$Longitude <- as.numeric(data$Longitude)
data$Latitude <- as.numeric(data$Latitude)
data <- unique(data)
IDs <- sort(unique(data$UnitId))
# cull on date
idx <- which(month(data$POSIX) > 3)
data <- data[idx,]
# cull on position - some a in Denmark ...
idx <- which(data$Latitude > 56)
data <- data[idx,]
# get the list of Station numbers vs station names
namesnumbers <- read.csv("DATA/names_numbers.txt",sep="",header=F)
namesnumbers[,1] <- as.character(namesnumbers[,1])

# Try a single 'cut-off' date for all units
# try end of July as last useful date
idx <- which(month(data$POSIX) < 8)
data <- data[idx,]
# try mid-July as last useful date
idx <- which(data$POSIX < "2022-07-10")
data <- data[idx,]
```

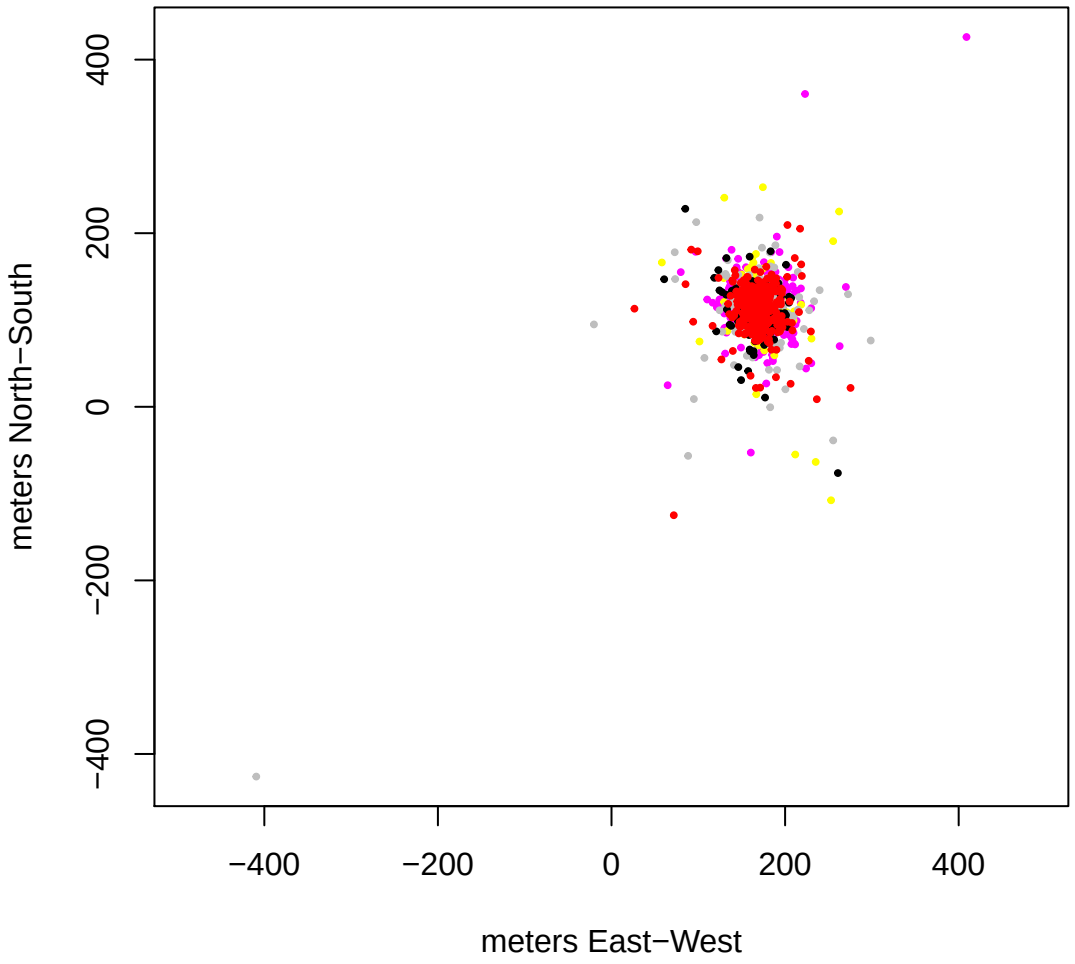
Utility GC formula

```
# Calculates the geodesic distance between two points specified by radian lat/lon using the
# Haversine formula (hf)
gcd.hf <- function(long1, lat1, long2, lat2) {
  R <- 6371*1000 # Earth mean radius [m]
  delta.long <- (long2 - long1)
  delta.lat <- (lat2 - lat1)
  a <- sin(delta.lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta.long/2)^2
  c <- 2 * asin(min(1,sqrt(a)))
  d = R * c
  return(d) # Distance in m
}
```

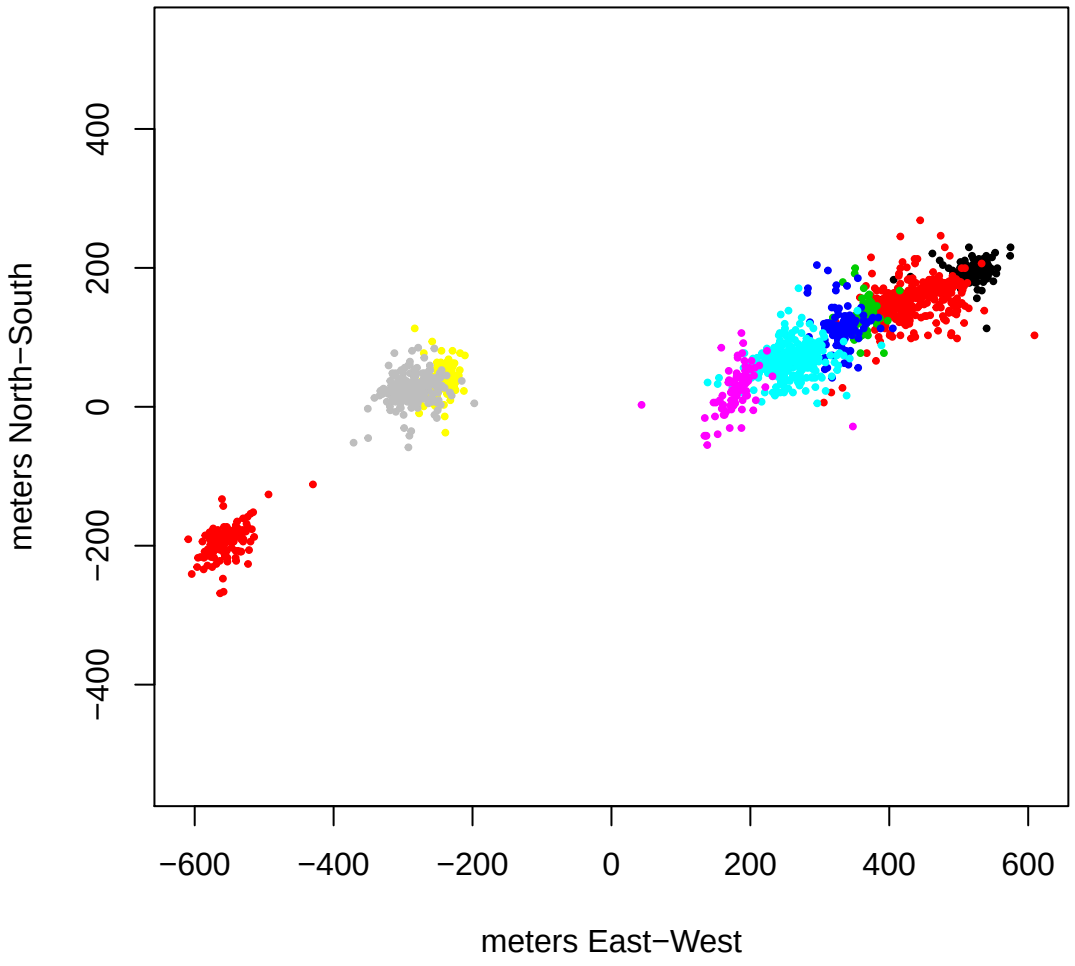
read and plot each file

```
# list the important times - start, jumps, ending:
important_times <- c(as.POSIXct("2022-03-31 00:00:00",tz="UTC"),as.POSIXct("2022-04-07 00:00:00",tz="UTC
                     as.POSIXct("2022-04-07 00:00:00",tz="UTC"),as.POSIXct("2022-04-24 12:00:00",tz="UTC
                     as.POSIXct("2022-04-24 12:00:00",tz="UTC"),as.POSIXct("2022-04-27 12:00:01",tz="UTC
                     as.POSIXct("2022-04-27 12:00:01",tz="UTC"),as.POSIXct("2022-05-03 03:00:00",tz="UTC
                     as.POSIXct("2022-05-03 03:00:00",tz="UTC"),as.POSIXct("2022-05-19 22:00:00",tz="UTC
                     as.POSIXct("2022-05-19 22:00:00",tz="UTC"),as.POSIXct("2022-06-09 02:00:00",tz="UTC
                     as.POSIXct("2022-06-09 02:00:00",tz="UTC"),as.POSIXct("2022-06-15 02:00:00",tz="UTC
                     as.POSIXct("2022-06-15 02:00:00",tz="UTC"),as.POSIXct("2022-06-25 00:00:00",tz="UTC
                     as.POSIXct("2022-06-25 00:00:00",tz="UTC"),as.POSIXct("2022-06-30 23:00:00",tz="UTC
                     as.POSIXct("2022-06-30 23:00:00",tz="UTC"),as.POSIXct("2022-07-14 02:00:00",tz="UTC
                     as.POSIXct("2022-07-14 02:00:00",tz="UTC"),as.POSIXct("2022-08-02 02:00:00",tz="UTC
                     as.POSIXct("2022-08-02 02:00:00",tz="UTC"),as.POSIXct("2023-08-02 02:00:00",tz="UTC
list_of_eventful_dates <- unique(sort(important_times))
limitdates <- NULL
for (it in seq(from=1,to=length(important_times),by=2))
{  limitdates <- rbind(limitdates,c(anytime(important_times[it],asUTC=T),anytime(important_times[it+1],a


for (ifil in IDs)
{
  name <- ifil
  print("-------------------------------------------")
  nom <- namesnumbers[which(namesnumbers[,2] == name),1]
  print(paste(" Processing file ",name," = ",nom))

  sdx <- which(data$UnitId == ifil)
  df <- data[sdx,]
  df <- na.omit(df)
  xy <- latlong2km(df$Longitude, df$Latitude)

  #plot(xy$km.e*1000,xy$km.n*1000,asp=1,main=namesnumbers[which(namesnumbers[,2] == name),1],type="p",p
  plot(xy$km.e*1000,xy$km.n*1000,asp=1,main=namesnumbers[which(namesnumbers[,2] == name),1],type="p",pch
  # overplot colours for each segment
  for (iseg in 1:nrow(limitdates))
  {
    idx <- which(df$POSIX >= limitdates[iseg,1] & df$POSIX < limitdates[iseg,2])
    points(xy$km.e[idx]*1000,xy$km.n[idx]*1000,col=iseg,pch=19,cex=0.4)
  } # end iseg loop
  print("-------------------------------------------")
} # end ifil loop
```

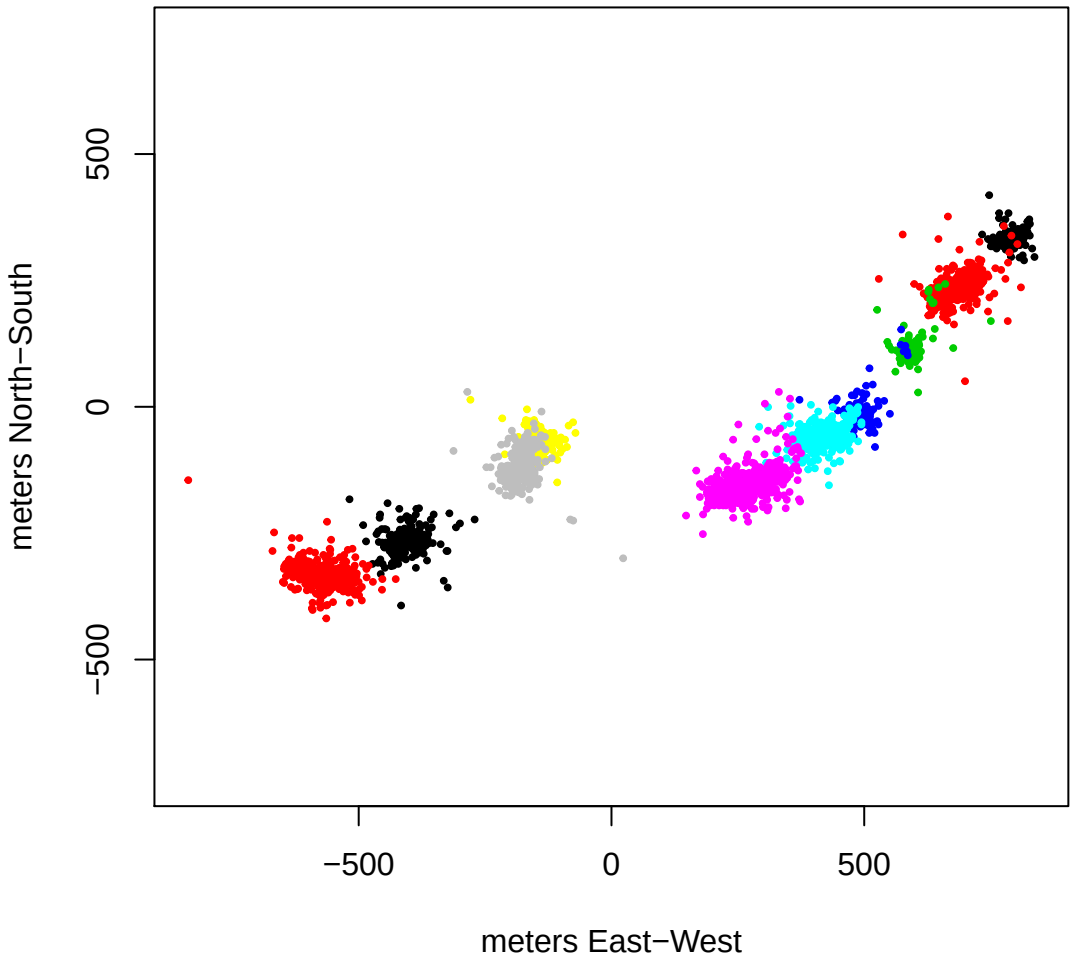
```
## [1] "-------------------------------------------"
## [1] " Processing file  81619  =  Fjeldrype"
```



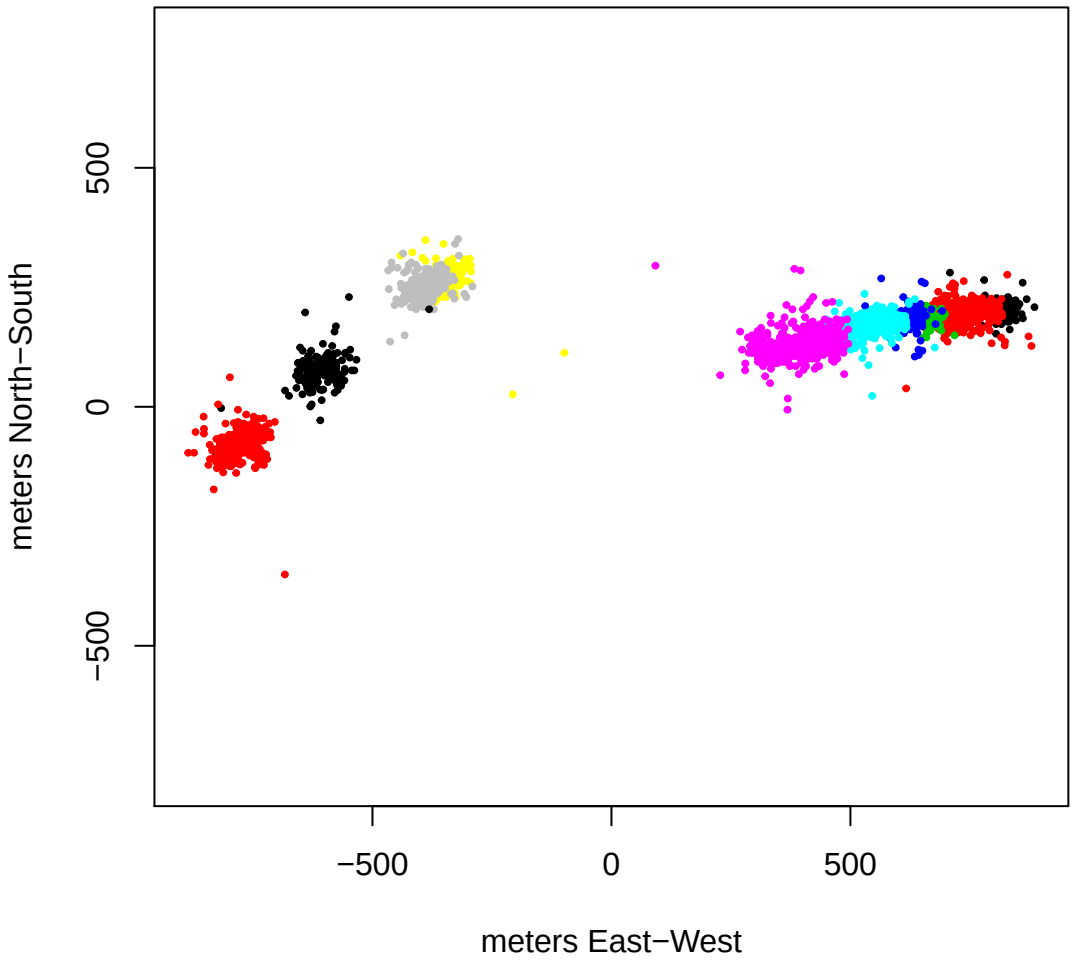

```
## [1] "-----------------------------------------"
## [1] "-----------------------------------------"
## [1] " Processing file  88319  =  Søkonge"
```



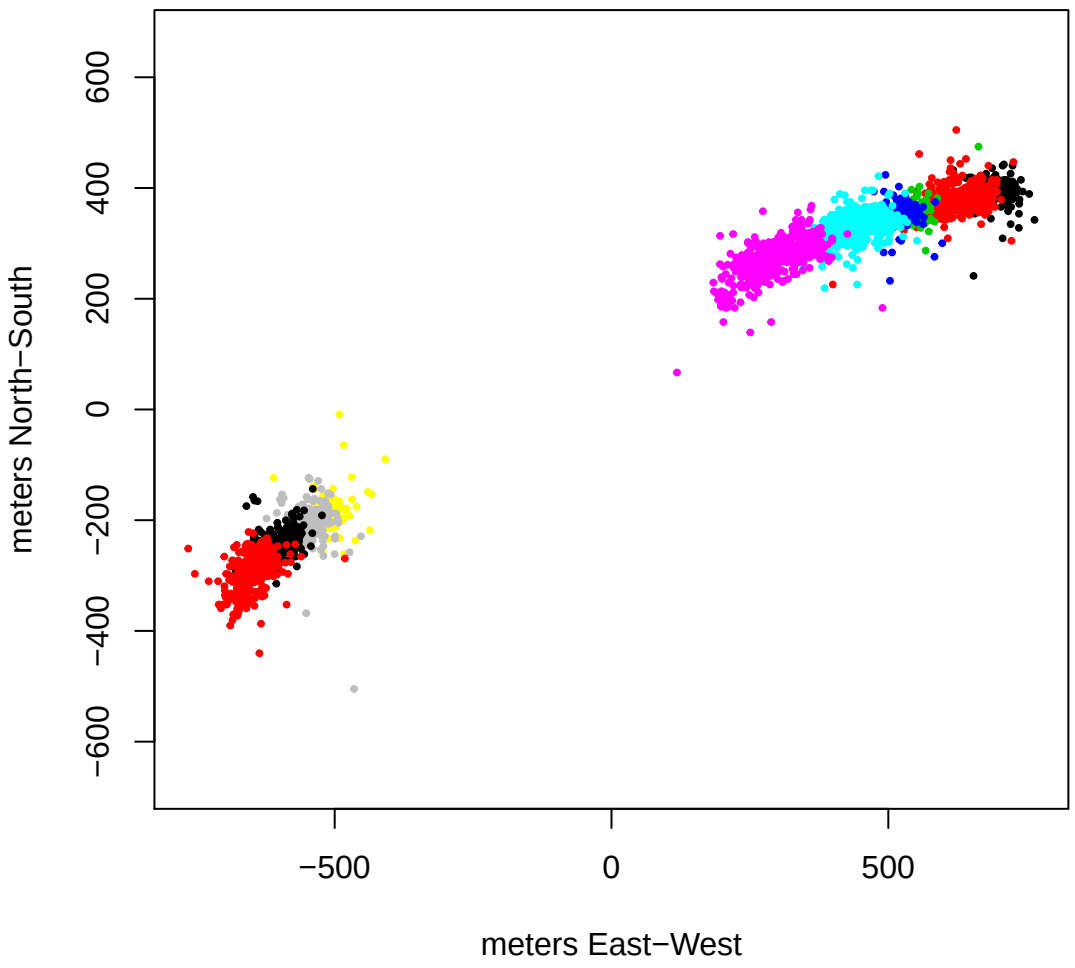

```
## [1] "----------------------------------------"
## [1] "----------------------------------------"
## [1] " Processing file  88462  =  Mallemuk"
```



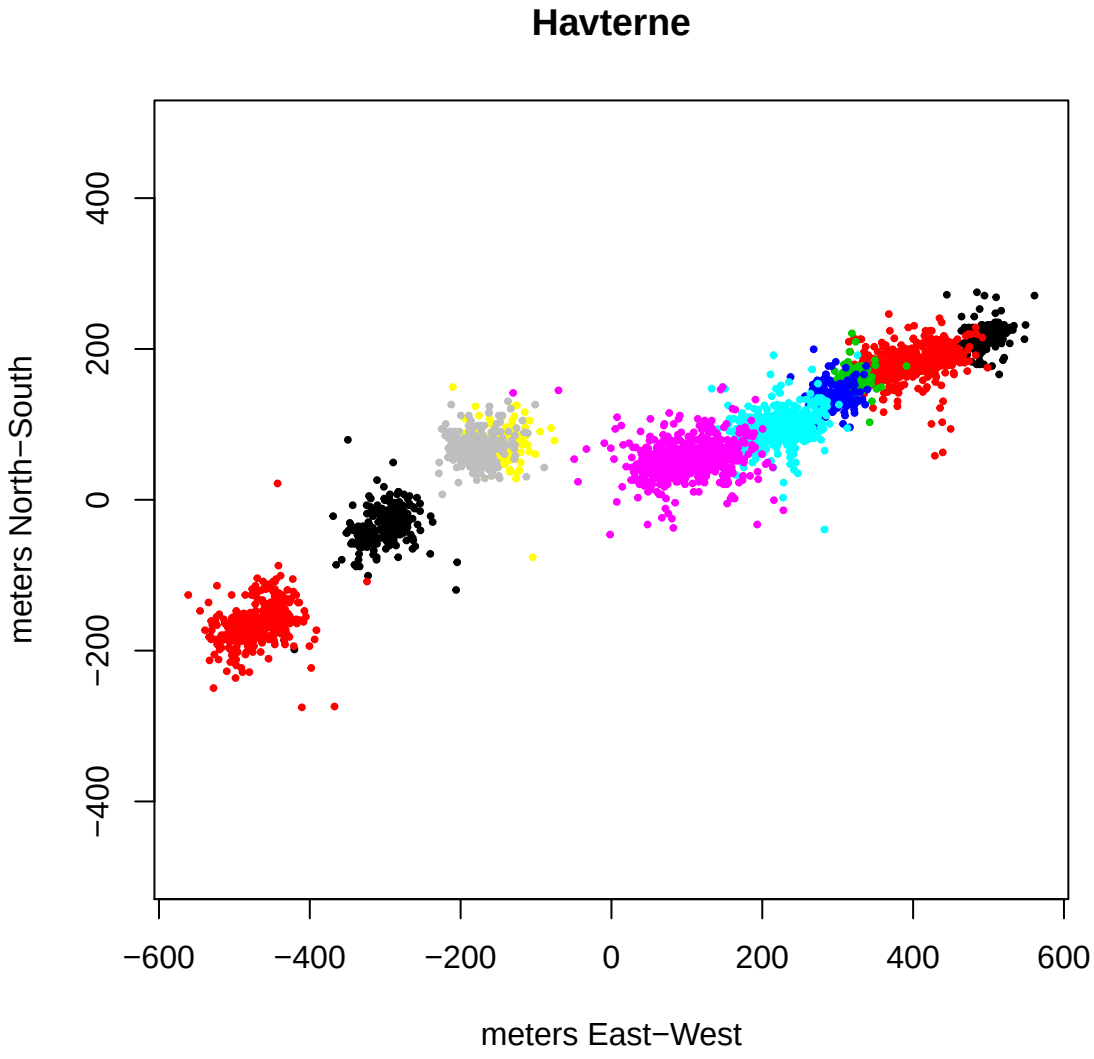

```
## [1] "----------------------------------------"
## [1] "----------------------------------------"
## [1] " Processing file  88463  =  Havørn"
```



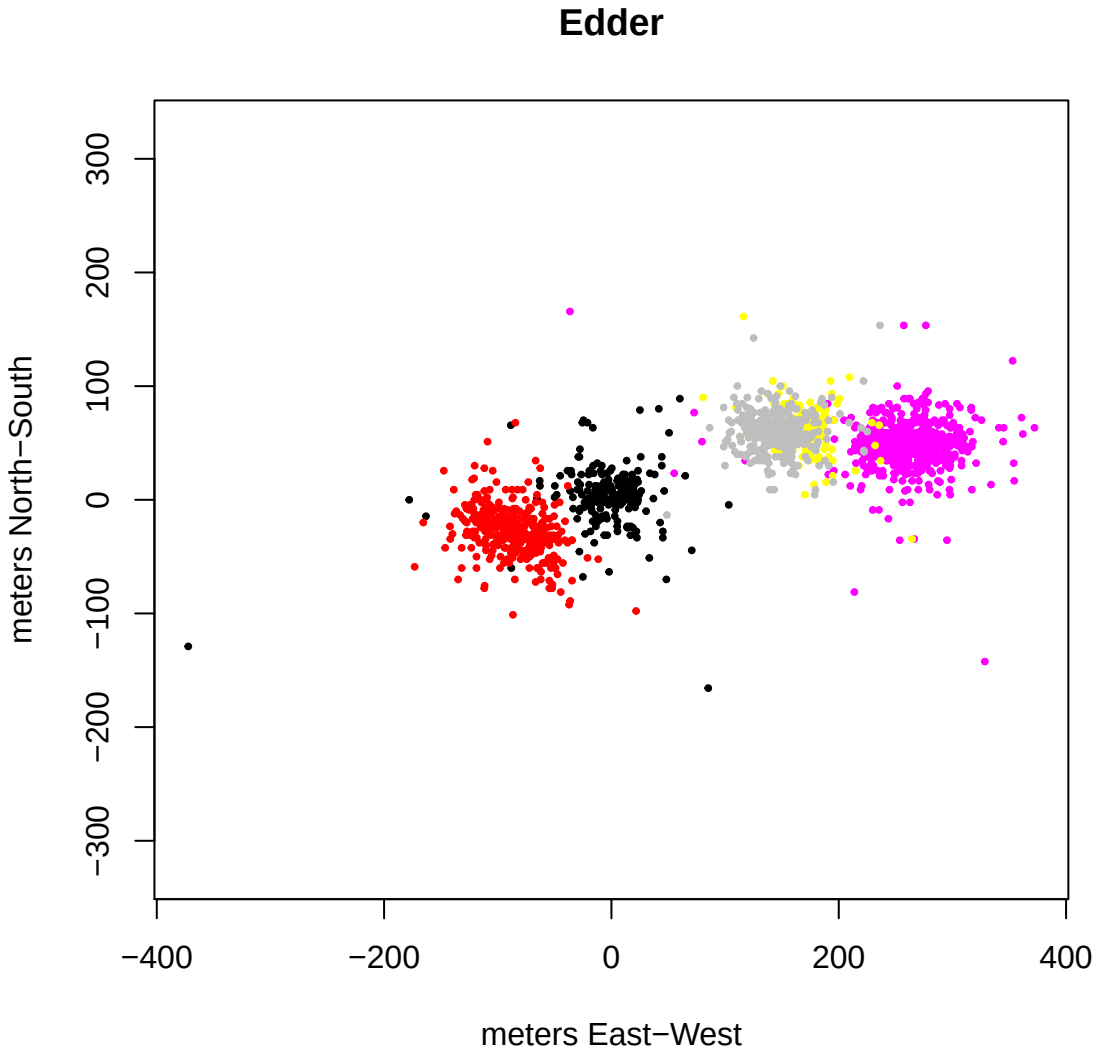

```
## [1] "----------------------------------------"
## [1] "----------------------------------------"
## [1] " Processing file  88464  =  Ismåge"
```




```
## [1] "-----------------------------------------"
## [1] "-----------------------------------------"
## [1] " Processing file  88465  =  Havterne"
```




```
## [1] "-----------------------------------------"
## [1] "-----------------------------------------"
## [1] " Processing file  88617  =  Edder"
```




```
## [1] "----------------------------------------"
```

Plot coloured points

```
plotcolouredpoints <- function(x,y,limitdates,ipair,ivar)
{
  idx <- which(df$UTC >= limitdates[ipair,1] & df$UTC < limitdates[ipair,2])
  points(x[idx],y[idx],type="p",cex=0.3,col=1+ipair)
}
```

Model motion 2

model positions and calculate speeds at jumps

```r
model_motion2 <- function(df, name, limitdates)
{
  par(mfrow = c(2, 1))
  nlimits <- nrow(limitdates)
  statname <-
    name #strsplit(strsplit(name, "/")[[1]][2],".rds")[[1]][1]
  # Latitude
  latitude_pred_at_interval_left_right <- NULL
  lat_speed <- NULL
  # loop over limidates and model positions before and after each limit date
  # plot(df$POSIX,df$Latitude,type="p",xlim=range(df$POSIX),ylim=range(df$Latitude,na.rm=T),xlab="Time"
  plot(
    df$POSIX,
    df$Latitude,
    type = "p",
    xlim = range(df$POSIX),
    ylim = range(df$Latitude, na.rm = T),
    xlab = "Time",
    ylab = "Latitude",
    main = namesnumbers[which(namesnumbers[, 2] == statname), 1],
    pch = 19,
    cex = 0.2
  )
  for (ilimit in 1:nlimits)
  {
    idx <-
      which(df$POSIX >= limitdates[ilimit, 1] &
              df$POSIX < limitdates[ilimit, 2] &
              !is.na(df$Latitude))
    if (length(idx) != 0) {
      rlmfit <- rlm(df$Latitude[idx] ~ df$POSIX[idx])
      lat_speed <-
        rbind.data.frame(lat_speed,
                         c(ilimit, summary(rlmfit)$coefficients[2] * 3600 / 180 * pi)) # radians per ho
      if (ilimit == 1) {
        #plot(df$POSIX[idx],df$Latitude[idx],type="p",xlim=range(df$POSIX),ylim=range(df$Latitude,na.rm
        points(df$POSIX[idx],
               df$Latitude[idx],
               pch = 19,
               cex = 0.2)
        lines(df$POSIX[idx],
              rlmfit$fitted.values,
              col = 2,
              lwd = 3)
        # evaluate diff at jump
        latitude_pred_at_interval_left_right  <-
          c(first(rlmfit$fitted.values),
            last(rlmfit$fitted.values))
      }
      if (ilimit  > 1) {
        points(df$POSIX[idx],
               df$Latitude[idx],
               pch = 19,
```

```
              cex = 0.2)
      lines(df$POSIX[idx],
            rlmfit$fitted.values,
            col = 2,
            lwd = 3)
      #
      latitude_pred_at_interval_left_right  <-
        rbind.data.frame(latitude_pred_at_interval_left_right,
                         c(
                           first(rlmfit$fitted.values),
                           last(rlmfit$fitted.values)
                         ))
    }
  } # end length(idx)
}
colnames(lat_speed) <- c("segment_number", "lat_speed_radperhr")
# Longitude
longitude_pred_at_interval_left_right <- NULL
lon_speed <- NULL
leftrightdates <- NULL
# loop over limidates and model positions before and after each limitdate
#plot(df$POSIX,df$Longitude,type="p",xlim=range(df$POSIX),ylim=range(df$Longitude,na.rm=T),xlab="Time
plot(
  df$POSIX,
  df$Longitude,
  type = "p",
  xlim = range(df$POSIX),
  ylim = range(df$Longitude, na.rm = T),
  xlab = "Time",
  ylab = "Longitude",
  main = namesnumbers[which(namesnumbers[, 2] == statname), 1],
  pch = 19,
  cex = 0.2
)
for (ilimit in 1:nlimits)
{
  idx <-
    which(df$POSIX >= limitdates[ilimit, 1] &
            df$POSIX < limitdates[ilimit, 2] &
            !is.na(df$Longitude))
  if (length(idx) != 0) {
    #print(c("f",length(idx)))
    rlmfit <- rlm(df$Longitude[idx] ~ df$POSIX[idx])
    lon_speed <-
      rbind.data.frame(lon_speed,
                       c(ilimit, summary(rlmfit)$coefficients[2] * 3600 / 180 * pi)) # radians per ho
    #print(c(rlmfit$fitted.values[1],last(rlmfit$fitted.values)))
    if (ilimit == 1) {
      #browser()
      #plot(df$POSIX[idx],df$Longitude[idx],type="p",xlim=range(df$POSIX),ylim=range(df$Longitude,na.
      points(df$POSIX[idx],
             df$Longitude[idx],
             pch = 19,
```

```
                cex = 0.2)
        lines(df$POSIX[idx],
              rlmfit$fitted.values,
              col = 2,
              lwd = 3)
        # evaluate diff at jump
        longitude_pred_at_interval_left_right  <-
          c(first(rlmfit$fitted.values),
            last(rlmfit$fitted.values))
      }
      if (ilimit  > 1) {
        points(df$POSIX[idx],
               df$Longitude[idx],
               pch = 19,
               cex = 0.2)
        lines(df$POSIX[idx],
              rlmfit$fitted.values,
              col = 2,
              lwd = 3)
        #
        longitude_pred_at_interval_left_right  <-
          rbind.data.frame(longitude_pred_at_interval_left_right,
                           c(
                             first(rlmfit$fitted.values),
                             last(rlmfit$fitted.values)
                           ))

        #leftrightdates <- rbind.data.frame(leftrightdates,c(limitdates[ilimit, 1],limitdates[ilimit, 2
        leftrightdates <- rbind(leftrightdates,as.POSIXct(limitdates[ilimit, ],origin="1970-01-01 00:00
        print(as.character(as.POSIXct(limitdates[ilimit, ],origin="1970-01-01 00:00:00",tz="UTC")))
        #browser()
      }
    }
  }
  colnames(lon_speed) <- c("segment_number", "lon_speed_radperhr")

  segment_speed <-
    Rearth * sqrt((lon_speed[, 2] * cos(median(
      df$Latitude, na.rm = T
    ) / 180 * pi)) ^ 2 + (lat_speed[, 2]) ^ 2) # meters per hour


  #browser()
  return(
    list(
      "lats" = latitude_pred_at_interval_left_right,
      "longs" = longitude_pred_at_interval_left_right,
      "segspeed" = segment_speed,
      "left_right_dates" = leftrightdates
    )

  )
}
```

function to get interval and jump speeds

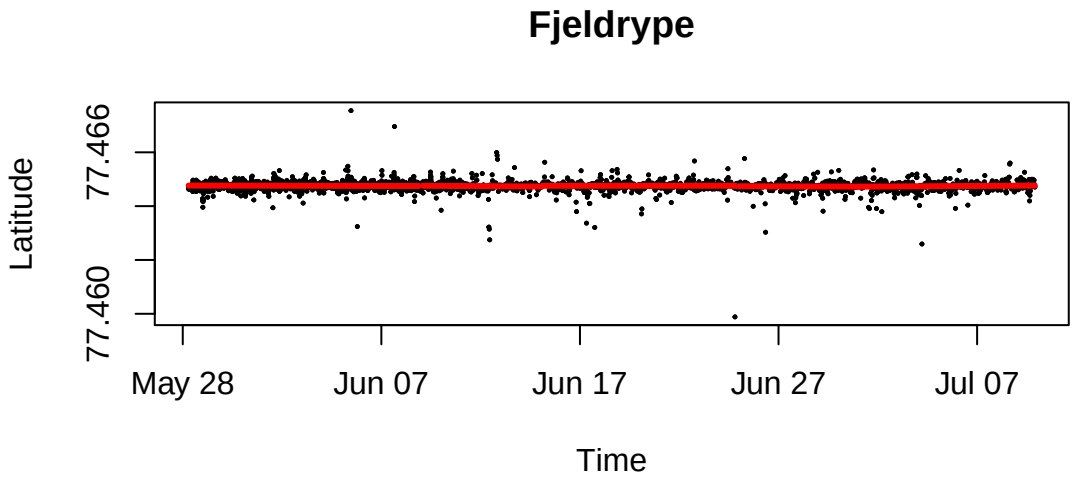
```
get_surge_speeds <- function(listerne,lon_in,lat_in)
{

  lon <- lon_in/180*pi # in radians
  lat <- lat_in/180*pi
  delta_t <- 1 # hours

  lon_here <- listerne$longs
  lat_here <- listerne$lats

  # calculate jump speeds
  n_segments <- nrow(listerne$lats)
  speed <- NULL
  for (iseg in 1:(n_segments-1))
  {
    delta_lon <- (lon_here[iseg,2]-lon_here[iseg+1,1])/180*pi
    delta_lat <- (lat_here[iseg,2]-lat_here[iseg+1,1])/180*pi

    speed <- rbind.data.frame(speed,c(iseg,Rearth/delta_t*sqrt(delta_lon^2*cos(lat)^2+delta_lat^2))) #
  }
  colnames(speed) <- c("jump_number","speed_metersph")
  #browser()
  return(list("speed_jump"=speed))
}
```

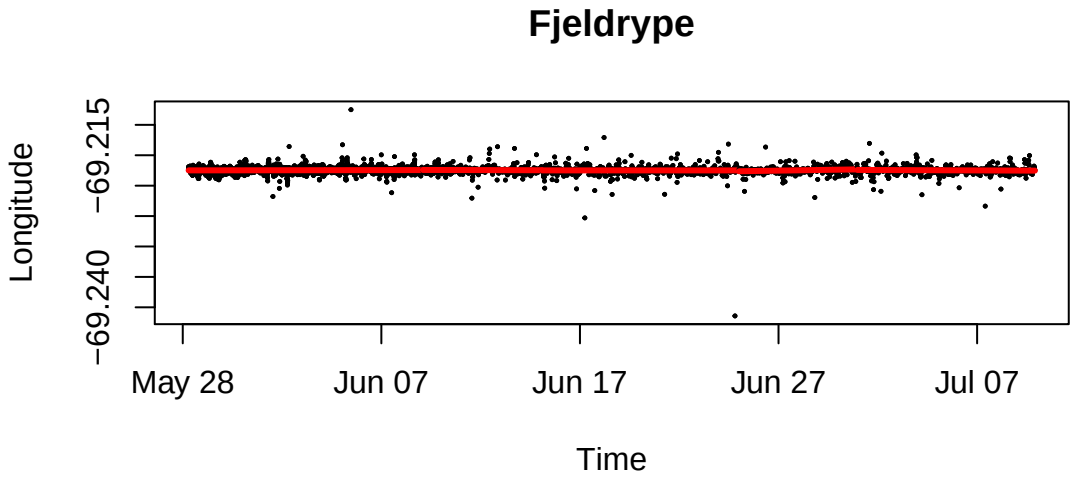
read and plot each file

```
for (id in IDs)

{
  print("-----------------------------------------")
  print(paste(" Processing unitID ",id," = ",namesnumbers[which(namesnumbers[,2] == id),1]))
  mdx <- which(data$UnitId == id)
  df <- data[mdx,]

  # model speed as unconnected straight line segments
  listerne <- model_motion2(df,id,limitdates)
  #browser()
  segspeeds <- round(listerne$segspeed,2)
  print("Segment speeds in m/hr : ")
  print(segspeeds)

  # get surge speeds
  surge_speeds <- get_surge_speeds(listerne,lon=median(df$Longitude,na.rm=T),lat=median(df$Latitude,na.
  print(surge_speeds)
  #browser()
  #
  for (ijk in 1:(length(segspeeds)-1))
  {
```

```
    print(c(ijk,unname(as.POSIXct(unlist(listerne$left_right_dates[ijk,]),origin="1970-01-01 00:00:00",
    print(c(ijk,unname(as.POSIXct(unlist(listerne$left_right_dates[ijk,]),origin="1970-01-01 00:00:00",
  }
  print(c(ijk,unname(as.POSIXct(unlist(listerne$left_right_dates[ijk,]),origin="1970-01-01 00:00:00",tz=
}
```

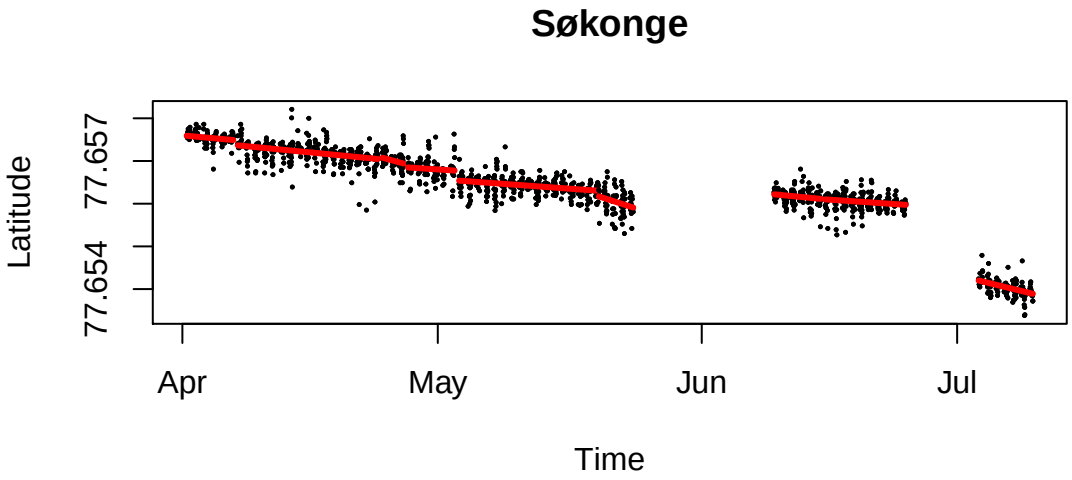
```
## [1] "-----------------------------------------"
## [1] " Processing unitID  81619  =  Fjeldrype"
```

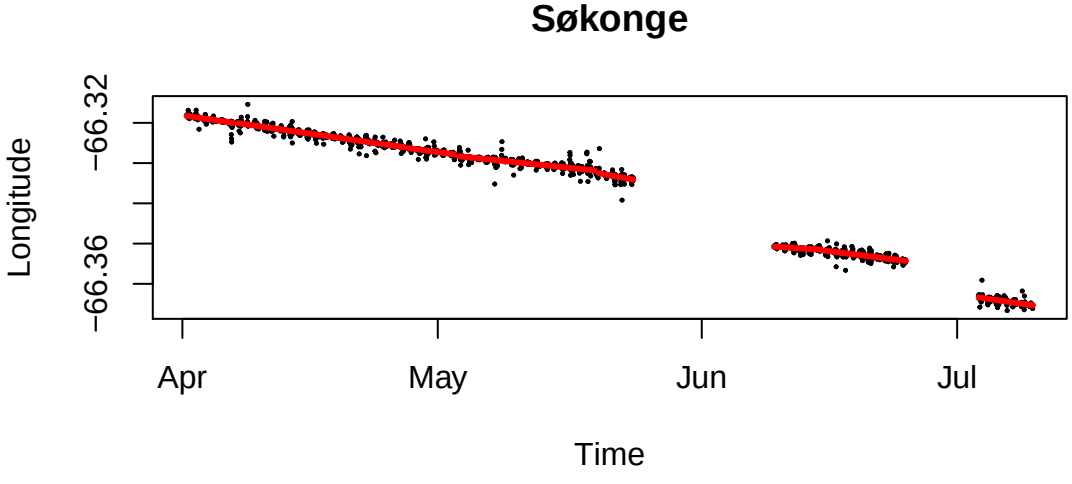





```
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
## [1] 0.01 0.02 0.00 0.05 0.02
## $speed_jump
##   jump_number speed_metersph
## 1           1       4.026615
## 2           2       1.781471
## 3           3       4.962987
```

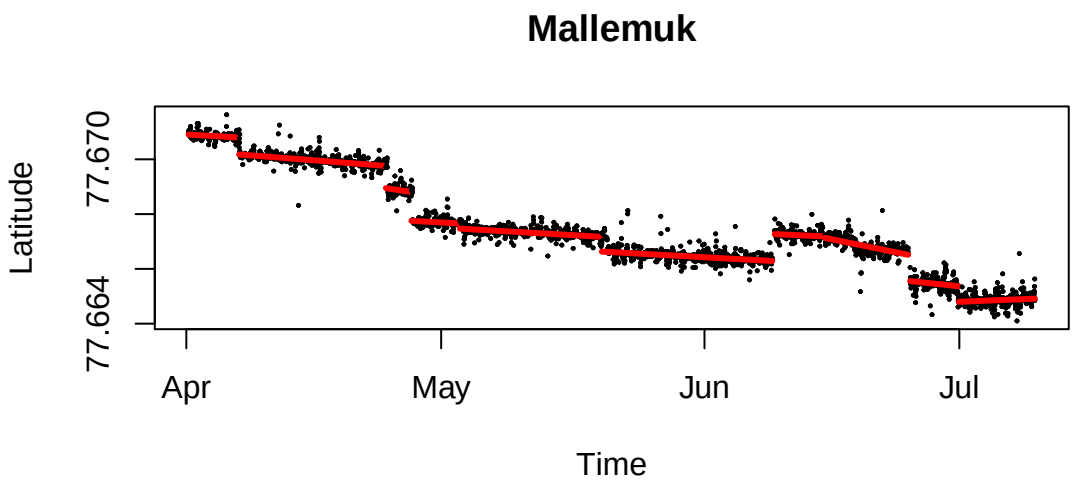
```
## 4           4       3.541576
##
## [1] 1.000000e+00 1.652998e+09 1.654740e+09 1.000000e-02
## [1] 1.000000e+00 1.652998e+09 1.654740e+09 4.026615e+00
## [1] 2.000000e+00 1.654740e+09 1.655258e+09 2.000000e-02
## [1] 2.000000e+00 1.654740e+09 1.655258e+09 1.781471e+00
## [1]          3 1655258400 1656115200          0
## [1] 3.000000e+00 1.655258e+09 1.656115e+09 4.962987e+00
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 5.000000e-02
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 3.541576e+00
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 2.000000e-02
## [1] "----------------------------------------"
## [1] " Processing unitID  88319  =  Søkonge"
```

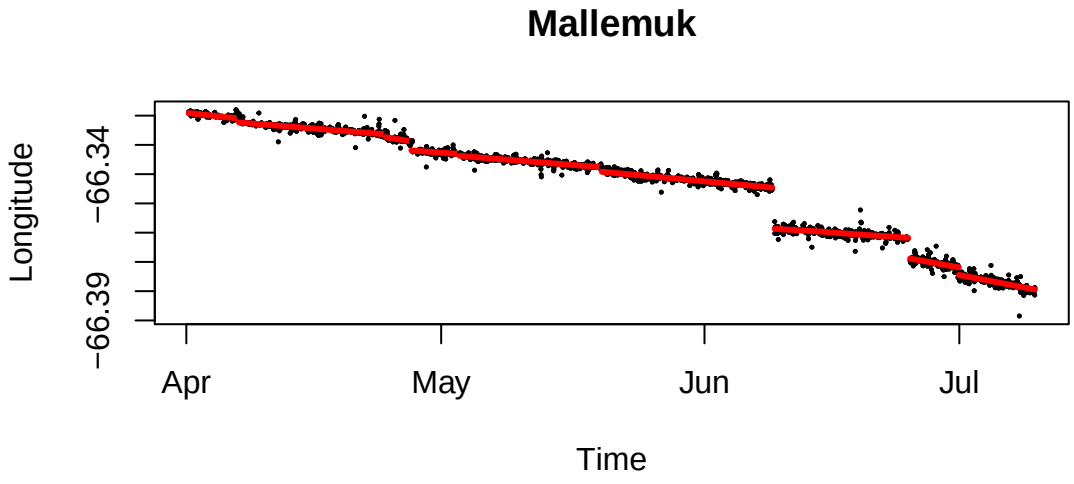





```
## [1] "2022-04-07 00:00:00" "2022-04-24 12:00:00"
## [1] "2022-04-24 12:00:00" "2022-04-27 12:00:01"
## [1] "2022-04-27 12:00:01" "2022-05-03 03:00:00"
## [1] "2022-05-03 03:00:00" "2022-05-19 22:00:00"
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
```

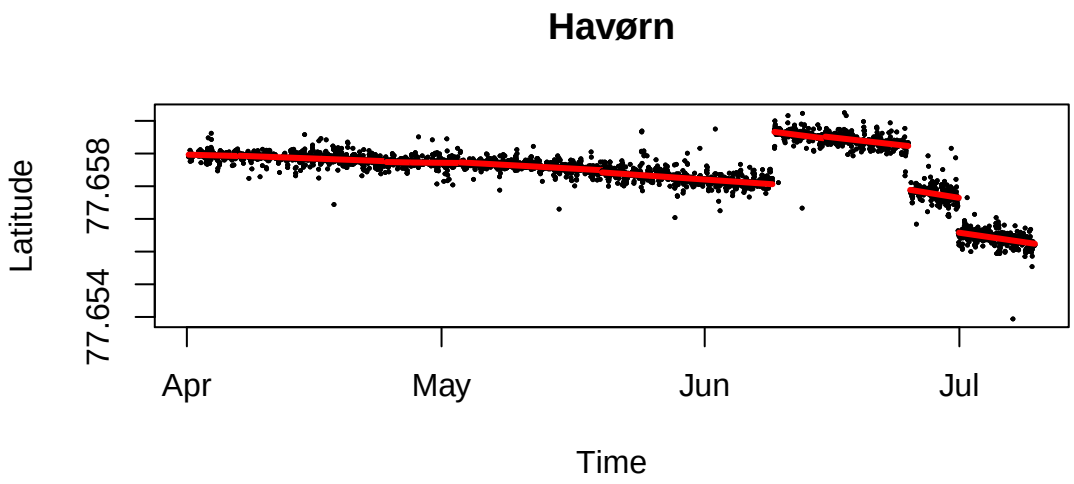
```
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
## [1] 0.32 0.32 0.39 0.29 0.23 0.49 0.16 0.27 0.38
## $speed_jump
##   jump_number speed_metersph
## 1           1      173.92024
## 2           2      148.86544
## 3           3       69.32710
## 4           4      147.60519
## 5           5      154.55195
## 6           6      451.07394
## 7           7       88.64938
## 8           8      405.53871
##
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 3.200000e-01
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 1.739202e+02
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 3.200000e-01
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 1.488654e+02
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 3.900000e-01
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 6.932710e+01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 2.900000e-01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 1.476052e+02
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.300000e-01
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 1.545519e+02
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 4.900000e-01
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 4.510739e+02
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.600000e-01
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 8.864938e+01
## [1] 8.000000e+00 1.656630e+09 1.657764e+09 2.700000e-01
## [1] 8.000000e+00 1.656630e+09 1.657764e+09 4.055387e+02
## [1] 8.000000e+00 1.656630e+09 1.657764e+09 3.800000e-01
## [1] "----------------------------------------"
## [1] " Processing unitID  88462  =  Mallemuk"
```

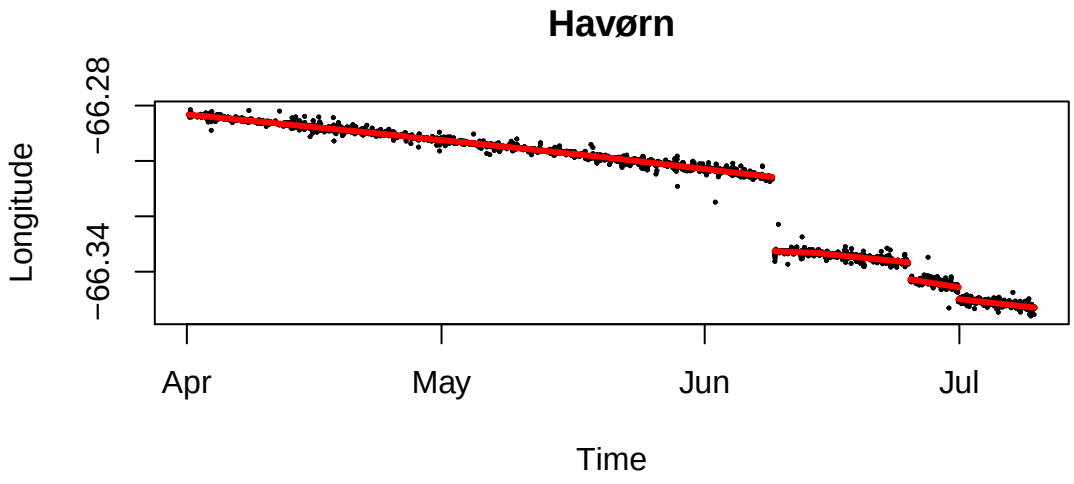





```
## [1] "2022-04-07 00:00:00" "2022-04-24 12:00:00"
## [1] "2022-04-24 12:00:00" "2022-04-27 12:00:01"
## [1] "2022-04-27 12:00:01" "2022-05-03 03:00:00"
## [1] "2022-05-03 03:00:00" "2022-05-19 22:00:00"
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
##  [1] 0.33 0.26 0.51 0.23 0.25 0.28 0.18 0.35 0.55 0.55
## $speed_jump
##   jump_number speed_metersph
## 1           1       214.5479
## 2           2       215.5812
## 3           3       197.1361
## 4           4       151.5021
## 5           5       289.2856
## 6           6       490.6172
## 7           7       111.3160
## 8           8       347.3525
```

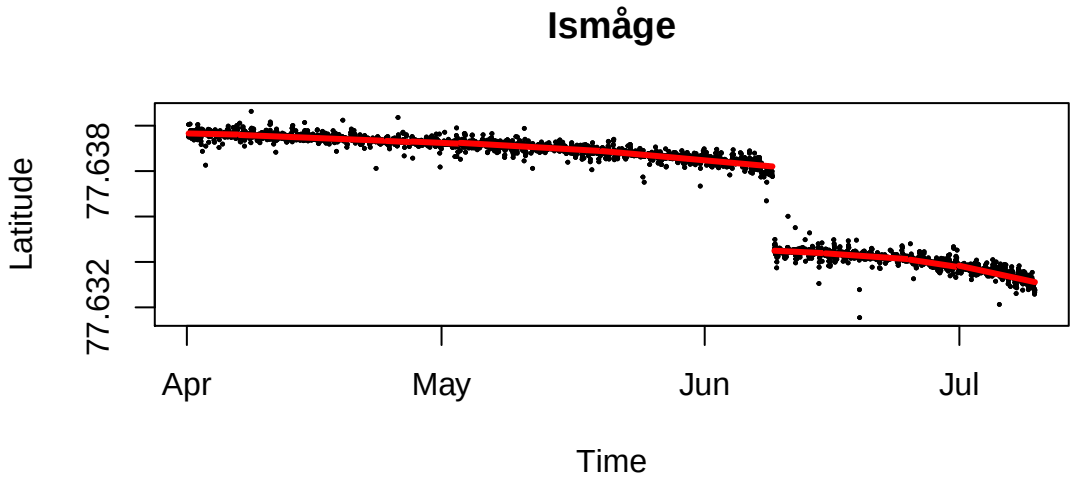
```
## 9           9       261.3492
##
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 3.300000e-01
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 2.145479e+02
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 2.600000e-01
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 2.155812e+02
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 5.100000e-01
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 1.971361e+02
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 2.300000e-01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 1.515021e+02
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.500000e-01
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.892856e+02
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 2.800000e-01
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 4.906172e+02
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.800000e-01
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.113160e+02
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 3.500000e-01
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 3.473525e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 5.500000e-01
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 2.613492e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 5.500000e-01
## [1] "----------------------------------------"
## [1] " Processing unitID  88463  =  Havørn"
```

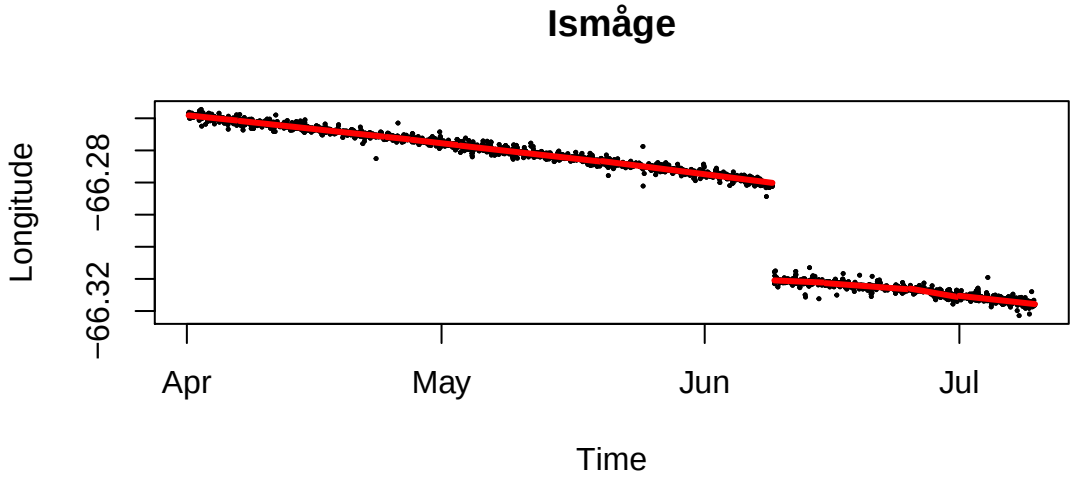





```
## [1] "2022-04-07 00:00:00" "2022-04-24 12:00:00"
## [1] "2022-04-24 12:00:00" "2022-04-27 12:00:01"
## [1] "2022-04-27 12:00:01" "2022-05-03 03:00:00"
## [1] "2022-05-03 03:00:00" "2022-05-19 22:00:00"
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
##  [1] 0.30 0.31 0.32 0.29 0.33 0.37 0.21 0.33 0.53 0.38
## $speed_jump
##   jump_number speed_metersph
## 1           1      169.33350
## 2           2      150.41437
## 3           3       64.25318
## 4           4      170.98435
## 5           5      309.75387
## 6           6      834.84610
## 7           7      107.20800
## 8           8      353.45532
```

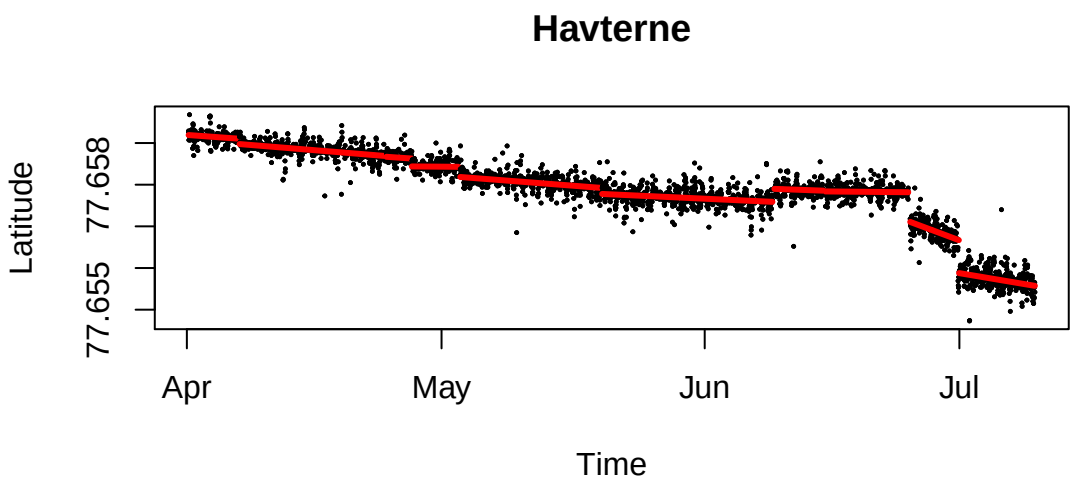
```
## 9           9      303.38328
##
## [1]          1.0 1649289600.0 1650801600.0          0.3
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 1.693335e+02
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 3.100000e-01
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 1.504144e+02
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 3.200000e-01
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 6.425318e+01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 2.900000e-01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 1.709843e+02
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 3.300000e-01
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 3.097539e+02
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 3.700000e-01
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 8.348461e+02
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 2.100000e-01
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.072080e+02
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 3.300000e-01
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 3.534553e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 5.300000e-01
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 3.033833e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 3.800000e-01
## [1] "----------------------------------------"
## [1] " Processing unitID  88464  =  Ismåge"
```

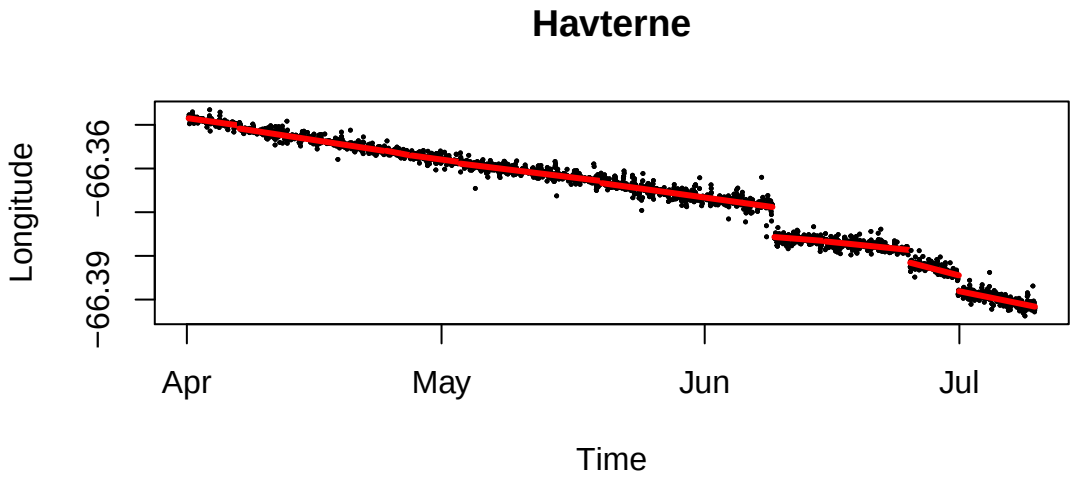





```
## [1] "2022-04-07 00:00:00" "2022-04-24 12:00:00"
## [1] "2022-04-24 12:00:00" "2022-04-27 12:00:01"
## [1] "2022-04-27 12:00:01" "2022-05-03 03:00:00"
## [1] "2022-05-03 03:00:00" "2022-05-19 22:00:00"
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
##  [1] 0.28 0.29 0.28 0.31 0.31 0.37 0.14 0.23 0.53 0.47
## $speed_jump
##   jump_number speed_metersph
## 1           1      161.74186
## 2           2      142.69024
## 3           3       62.05682
## 4           4      168.90422
## 5           5      299.02923
## 6           6     1029.00080
## 7           7       77.31835
```

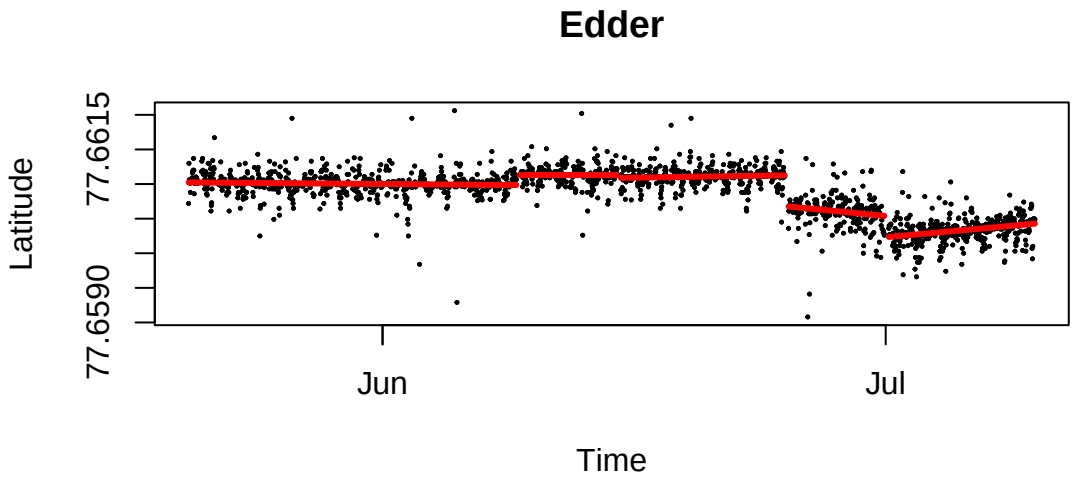
```
## 8           8      123.99466
## 9           9      158.51732
##
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 2.800000e-01
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 1.617419e+02
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 2.900000e-01
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 1.426902e+02
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 2.800000e-01
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 6.205682e+01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 3.100000e-01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 1.689042e+02
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 3.100000e-01
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.990292e+02
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 3.700000e-01
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 1.029001e+03
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.400000e-01
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 7.731835e+01
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 2.300000e-01
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 1.239947e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 5.300000e-01
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 1.585173e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 4.700000e-01
## [1] "----------------------------------------"
## [1] " Processing unitID  88465  =  Havterne"
```

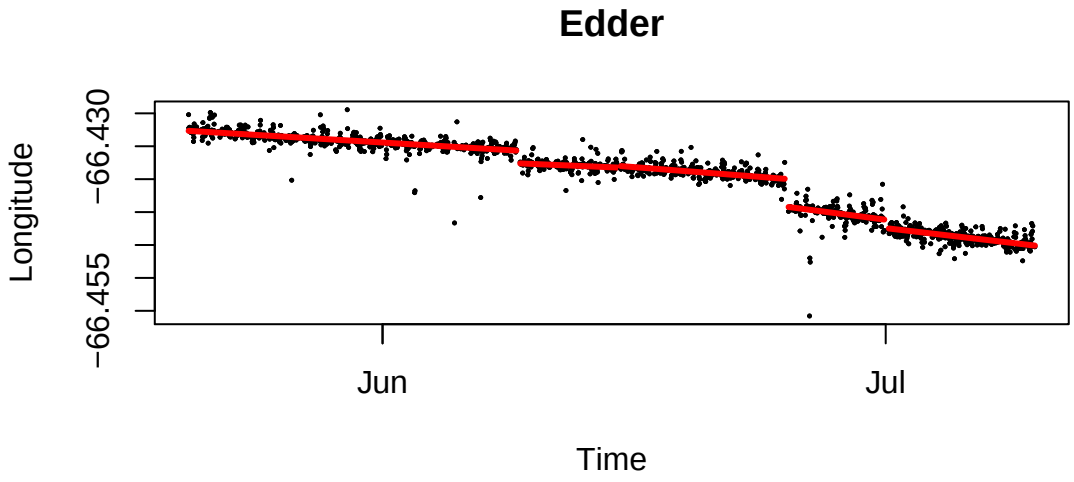





```
## [1] "2022-04-07 00:00:00" "2022-04-24 12:00:00"
## [1] "2022-04-24 12:00:00" "2022-04-27 12:00:01"
## [1] "2022-04-27 12:00:01" "2022-05-03 03:00:00"
## [1] "2022-05-03 03:00:00" "2022-05-19 22:00:00"
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
##  [1] 0.28 0.31 0.37 0.26 0.25 0.27 0.15 0.20 0.61 0.42
## $speed_jump
##   jump_number speed_metersph
## 1           1      187.53479
## 2           2      151.99716
## 3           3       65.96237
## 4           4      150.58439
## 5           5      245.31106
## 6           6      314.01080
## 7           7       70.37036
## 8           8      225.82996
```

```
## 9           9      293.36132
##
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 2.800000e-01
## [1] 1.000000e+00 1.649290e+09 1.650802e+09 1.875348e+02
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 3.100000e-01
## [1] 2.000000e+00 1.650802e+09 1.651061e+09 1.519972e+02
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 3.700000e-01
## [1] 3.000000e+00 1.651061e+09 1.651547e+09 6.596237e+01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 2.600000e-01
## [1] 4.000000e+00 1.651547e+09 1.652998e+09 1.505844e+02
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.500000e-01
## [1] 5.000000e+00 1.652998e+09 1.654740e+09 2.453111e+02
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 2.700000e-01
## [1] 6.000000e+00 1.654740e+09 1.655258e+09 3.140108e+02
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 1.500000e-01
## [1] 7.000000e+00 1.655258e+09 1.656115e+09 7.037036e+01
## [1]          8.0 1656115200.0 1656630000.0          0.2
## [1] 8.000000e+00 1.656115e+09 1.656630e+09 2.258300e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 6.100000e-01
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 2.933613e+02
## [1] 9.000000e+00 1.656630e+09 1.657764e+09 4.200000e-01
## [1] "----------------------------------------"
## [1] " Processing unitID  88617  =  Edder"
```






```
## [1] "2022-05-19 22:00:00" "2022-06-09 02:00:00"
## [1] "2022-06-09 02:00:00" "2022-06-15 02:00:00"
## [1] "2022-06-15 02:00:00" "2022-06-25 00:00:00"
## [1] "2022-06-25 00:00:00" "2022-06-30 23:00:00"
## [1] "2022-06-30 23:00:00" "2022-07-14 02:00:00"
## [1] "Segment speeds in m/hr : "
## [1] 0.15 0.11 0.20 0.35 0.31
## $speed_jump
##   jump_number speed_metersph
## 1           1      132.88157
## 2           2       55.49533
## 3           3      202.41068
## 4           4      141.99801
##
## [1] 1.000000e+00 1.652998e+09 1.654740e+09 1.500000e-01
## [1] 1.000000e+00 1.652998e+09 1.654740e+09 1.328816e+02
## [1] 2.000000e+00 1.654740e+09 1.655258e+09 1.100000e-01
## [1] 2.000000e+00 1.654740e+09 1.655258e+09 5.549533e+01
## [1]          3.0 1655258400.0 1656115200.0          0.2
## [1] 3.000000e+00 1.655258e+09 1.656115e+09 2.024107e+02
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 3.500000e-01
```

```
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 1.419980e+02
## [1] 4.000000e+00 1.656115e+09 1.656630e+09 3.100000e-01
```

```
print("-----------------------------------------")
```

```
## [1] "-----------------------------------------"
```

Summary of eventful dates

```
list_of_eventful_dates
```

```
##  [1] "2022-03-31 00:00:00 UTC" "2022-04-07 00:00:00 UTC"
##  [3] "2022-04-24 12:00:00 UTC" "2022-04-27 12:00:01 UTC"
##  [5] "2022-05-03 03:00:00 UTC" "2022-05-19 22:00:00 UTC"
##  [7] "2022-06-09 02:00:00 UTC" "2022-06-15 02:00:00 UTC"
##  [9] "2022-06-25 00:00:00 UTC" "2022-06-30 23:00:00 UTC"
## [11] "2022-07-14 02:00:00 UTC" "2022-08-02 02:00:00 UTC"
## [13] "2023-08-02 02:00:00 UTC"
```