The heat is on: Rising temperatures alter when, how, and where large terrestrial mammals move

Appendix A: Cleaning Telemetry Data

Stefano Mezzini[1,2] Chris H. Fleming[3,4] Siobhan Darlington[1,2]
Adam T. Ford[1,2] TJ Gooliaff[5] Karen E. Hodges[1,2] Kirk Safford[6]
Robert Serrouya[1,2,7] Michael J. Noonan[1,2,8]

[1] Okanagan Institute for Biodiversity, Resilience, and Ecosystem Services, The University of British Columbia Okanagan, Kelowna, British Columbia, Canada.

[2] Department of Biology, The University of British Columbia Okanagan, Kelowna, British Columbia, Canada.

[3] Department of Biology, University of Central Florida, Orlando, Florida 32816, United States.

[4] Smithsonian Conservation Biology Institute, National Zoological Park, 1500 Remount Rd., Front Royal, VA 22630, United States.

[5] British Columbia Ministry of Forests, Lands, Natural Resource Operations and Rural Development, Penticton, BC, Canada.

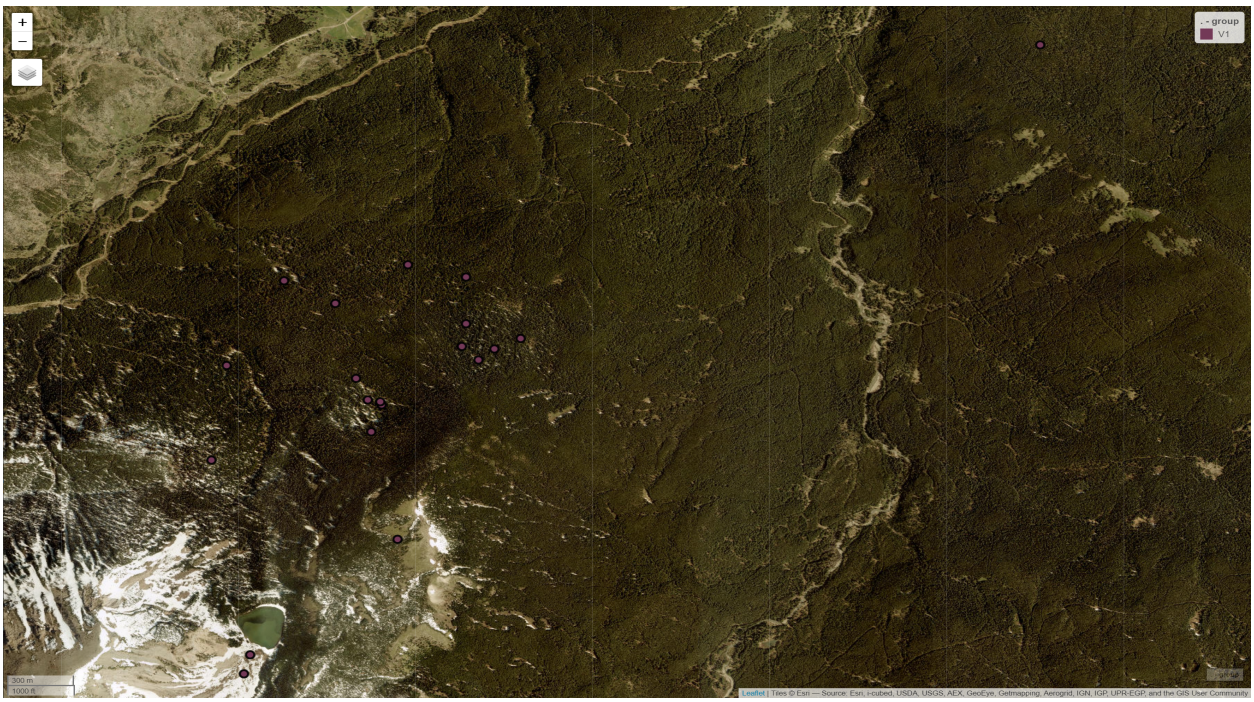
[6] British Columbia Ministry of Environment and Parks, Penticton, BC, Canada.

[7] Wildlife Science Centre, Biodiversity Pathways, University of British Columbia Okanagan, Revelstoke, British Columbia, Canada.

[8] Department of Computer Science, Math, Physics, and Statistics, The University of British

Columbia Okanagan, Kelowna, British Columbia, Canada.

In this appendix, we illustrate the cleaning procedure for the telemetry data. For the sake of brevity, we only show the code and outputs for some elk (*Cervus canadensis*); the full script is available at github.com/QuantitativeEcologyLab/bc-mammals-temperature/blob/main/analysis/02-remove-outliers.R. We suggest looking at the script since this appendix does not include the interactive maps that are generated by `plot_adj(..., map = TRUE)`. The interactive maps often show linear features (e.g., roads) and other geographic features (e.g., mountain ridges, rivers) that were useful to see during the data cleaning. Therefore, we have left the function calls in the document for completeness, but we have commented them out to prevent them from being executed. Below is an example of a map with some locations for elk E003, as generated by running `plot_adj('E006', max_speed = 0.4, max_angle = 170, map = TRUE)`.



We start by attaching the necessary packages and sourcing some custom functions. The custom functions are all available on GitHub in the `functions` folder of the repository located at github.com/QuantitativeEcologyLab/bc-mammals-temperature. While this appendix only shows the removal of a single outlier point, the appendix details the approach we used to

clean the telemetry data. We removed outlier points using `flag_outlier(..., value = 1)`, which leverages `ctmm::as.telemetry()` to remove rows that have `TRUE` or a positive, non-zero number in the `outlier` column. See the comments in each code chunk for additional detail and considerations about individual data points.

```
library('dplyr')     # for data wrangling (mutate(), %>%, etc.)
library('tidyr')     # for data wrangling (nest(), unnest(), pivot_*, etc.)
library('purrr')     # for functional programming (map_***(), etc.)
library('ctmm')      # for movement models
library('lubridate') # for working with dates
library('mapview')   # for interactive maps
library('ggplot2')   # for fancy plots
theme_set(theme_bw())

# source custom functions
source('functions/outlier_plots.R') # to plot outlier diagnostic plots
source('functions/check_animal.R') # to run diagnostic plots
source('functions/plot_adj.R') # to plot n adjacent locations
source('functions/flag_outlier.R') # to mark outliers
source('functions/remove_outlier_flags.R') # to start over with an animal
```

Before cleaning the telemetry data, we do some basic checks:

```
# dataset checks ----
# some NA timestamps (not enough to make a difference)
readRDS('data/tracking-data/all-tracking-data-not-cleaned.rds') %>%
  filter(is.na(timestamp)) %>%
  group_by(dataset_name) %>%
  summarize(n = n())
```

```
## # A tibble: 4 x 2
##   dataset_name                            n
##   <chr>                               <int>
## 1 Canis_lupus_boreal                      7
## 2 Elk in southwestern Alberta            37
## 3 Rangifer_tarandus_boreal               44
## 4 Rangifer_tarandus_southern_mountain     5
```

```
# ensure all only have one column of HDOP
readRDS('data/tracking-data/all-tracking-data-not-cleaned.rds') %>%
  filter(! is.na(timestamp)) %>%
  mutate(outlier = if_else(outlier, 1, 0)) %>%
  filter(! is.na(hdop) + ! is.na(HDOP) > 1) %>%
  nrow()
```

```
## [1] 0
```

```r
# should only have one column of DOP
readRDS('data/tracking-data/all-tracking-data-not-cleaned.rds') %>%
  filter(! is.na(timestamp)) %>%
  mutate(outlier = if_else(outlier, 1, 0)) %>%
  filter(! is.na(DOP) + ! is.na(gps.dop) > 1) %>%
  nrow()
```

```
## [1] 0
```

We can now import the data...

```r
# import the full dataset ----
# import the dataset after dropping NAs
d <- readRDS('data/tracking-data/all-tracking-data-not-cleaned.rds') %>%
  filter(! is.na(timestamp)) %>%
  mutate(hdop = if_else(is.na(hdop), HDOP, hdop), # merge error columns
         dop = if_else(is.na(DOP), gps.dop, DOP),
         original_outliers = outlier,
         outlier = if_else(outlier, 1, 0)) %>% # make numeric
  select(! c(HDOP, DOP, gps.dop)) %>% # remove duplicate columns
  relocate(dop, .after = pdop) %>%
  nest(tel = ! c(species, dataset_name, animal))
d # nested tibble of tracking datasets
```

```
## # A tibble: 434 x 4
##    animal species     dataset_name       tel
##    <chr>  <chr>       <chr>              <list>
##  1 BW003  Canis lupus Canis_lupus_boreal <tibble [15 x 16]>
##  2 BW006  Canis lupus Canis_lupus_boreal <tibble [61 x 16]>
##  3 BW008  Canis lupus Canis_lupus_boreal <tibble [11,846 x 16]>
##  4 BW009  Canis lupus Canis_lupus_boreal <tibble [9,153 x 16]>
##  5 BW010  Canis lupus Canis_lupus_boreal <tibble [11,952 x 16]>
##  6 BW012  Canis lupus Canis_lupus_boreal <tibble [11,627 x 16]>
##  7 BW013  Canis lupus Canis_lupus_boreal <tibble [5,298 x 16]>
##  8 BW014  Canis lupus Canis_lupus_boreal <tibble [11,710 x 16]>
##  9 BW015  Canis lupus Canis_lupus_boreal <tibble [5,629 x 16]>
## 10 BW016  Canis lupus Canis_lupus_boreal <tibble [4,914 x 16]>
## # i 424 more rows
```

... and do some last few check before cleaning.

```r
# ensure animal IDs are unique
sum(duplicated(d$animal))
```

```
## [1] 0
```

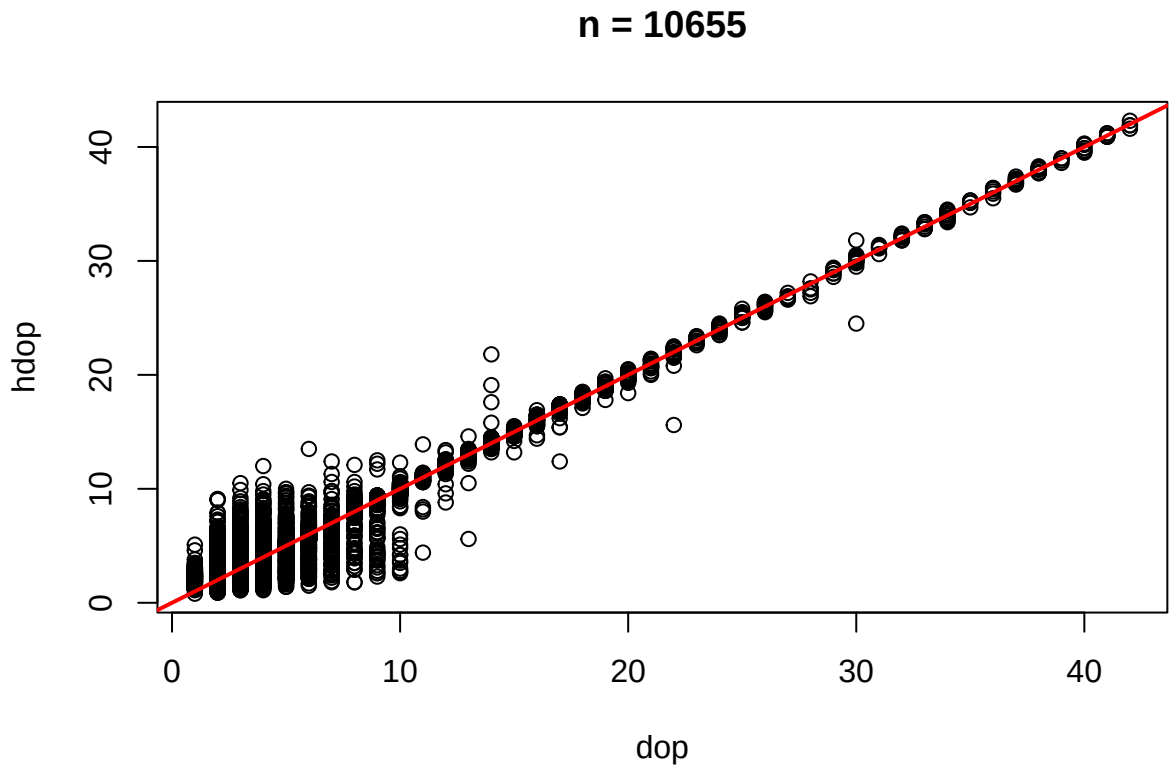
```r
# check which species have DOP values
d %>%
  unnest(tel) %>%
  group_by(species) %>%
  summarise(across(c(dop, hdop, pdop), ~ sum(.x, na.rm = TRUE))) %>%
  mutate(has_dop = dop + hdop + pdop > 0) %>%
  arrange(desc(has_dop))
```

```
## # A tibble: 6 x 5
##   species                     dop    hdop    pdop has_dop
##   <chr>                     <dbl>   <dbl>   <dbl> <lgl>
## 1 Canis lupus                   0  518510.      0  TRUE
## 2 Cervus canadensis       3060352.      0       0  TRUE
## 3 Oreamnos americanus           0     135. 108919. TRUE
## 4 Rangifer tarandus         87399  425988.      0  TRUE
## 5 Puma concolor                 0       0       0  FALSE
## 6 Ursus arctos horribilis       0       0       0  FALSE
```

```r
#' see`finite`: some SM caribou have different values for hdop and dop
#' other datasets are ok
d %>%
  unnest(tel) %>%
  mutate(has_hdop = ! is.na(hdop),
         has_dop = ! is.na(dop),
         has_pdop = ! is.na(pdop)) %>%
  group_by(dataset_name, has_hdop, has_dop, has_pdop) %>%
  summarise(hdop = mean(hdop, na.rm = TRUE),
            pdop = mean(pdop, na.rm = TRUE),
            dop = mean(dop, na.rm = TRUE)) %>%
  mutate(finite = map_int(1:n(),
                          ~ sum(! is.nan(c(hdop[.x], pdop[.x], dop[.x])))))
```

```
## # A tibble: 12 x 8
## # Groups:   dataset_name, has_hdop, has_dop [12]
##    dataset_name            has_hdop has_dop has_pdop   hdop   pdop    dop finite
##    <chr>                   <lgl>    <lgl>   <lgl>     <dbl>  <dbl>  <dbl>  <int>
##  1 Canis_lupus_boreal      TRUE     FALSE   FALSE      2.56 NaN    NaN        1
##  2 Elk in southwestern Al~ FALSE    FALSE   FALSE    NaN    NaN    NaN        0
##  3 Elk in southwestern Al~ FALSE    TRUE    FALSE    NaN    NaN      3.89     1
##  4 Oreamnos_americanus     FALSE    FALSE   TRUE     NaN      1.64 NaN        1
##  5 Oreamnos_americanus     TRUE     FALSE   FALSE      2.41 NaN    NaN        1
##  6 Puma_concolor_2         FALSE    FALSE   FALSE    NaN    NaN    NaN        0
##  7 Puma_concolor_4         FALSE    FALSE   FALSE    NaN    NaN    NaN        0
##  8 Rangifer_tarandus_bore~ TRUE     FALSE   FALSE      1.88 NaN    NaN        1
##  9 Rangifer_tarandus_sout~ FALSE    FALSE   FALSE    NaN    NaN    NaN        0
## 10 Rangifer_tarandus_sout~ FALSE    TRUE    FALSE    NaN    NaN      4.34     1
## 11 Rangifer_tarandus_sout~ TRUE     TRUE    FALSE      4.59 NaN      4.54     2
## 12 Ursus_arctos_horribilis FALSE    FALSE   FALSE    NaN    NaN    NaN        0
```

```
d %>%
  filter(dataset_name == 'Rangifer_tarandus_southern_mountain') %>%
  unnest(tel) %>%
  filter(! is.na(hdop) & ! is.na(dop)) %>%
  plot(hdop ~ dop, ., main = paste('n =', nrow(.)))
abline(a = 0, b = 1, col = 'red', lwd = 2)
```



Now that we have organized the full telemetry dataset, we can start cleaning the data. Locations are candidates for removal if they are far from the median location (i.e., large median deviation) or high minimum speed between locations. The minimum speed is calculated as the distance between locations divided by time between locations, since traveling in a straight line is the fastest possible path. To search for potential outliers, the `check_animal()` function creates five plots: The top plot shows the telemetry data with an appropriate two-point equidistant projection, and it can help detect any migrations or range shifts. The second plot, created by `ctmm::outlie(tel)` (where `tel` is a `telemetry` object, see Calabrese *et al.*, 2016) shows the telemetry data using a two-point equidistant projection. Points are larger

and redder if they are further from the median location, while segments are thicker and bluer if the minimum speed is large. Points that are near the median location and segments with small minimum speed may not be visible, while severe location errors will often be identifiable due to their very red points and thick blue segments (Fig. 1). The third plot, created by `plot(ctmm::outlie(tel))`, shows the minimum speed against median deviation. This plot is simply an alternative visualization of the second plot (without the spatial information). The fourth plot shows minimum speed against turning angle, since severe location errors will have high minimum speed and very sharp (i.e., large) turning angles (Fig. 1). The fifth plot shows minimum speed against sampling interval, which can be helpful to detect points that imply excessively high minimum speeds for long periods of time. Generally, animals tend to move faster (for long periods of time) when moving in a straight line (i.e., ballistic movement as opposed to exploratory movement).

Ideally, the second plot (center-left) should show a good mix of points and lines of different sizes, with no points or lines clearly standing out. Plots 3-5 should show a decrease in y values (vertical axis) as the x values (horizontal axis) increase. The intensity of the decrease will depend on a series of factors, including the animal's species and age. See elk E006 below for an example of a telemetry dataset with a clear GPS error that exhibits

In the two bottom figures produced by `check_animal()`, where minimum speed is plotted against turning angle and sampling interval, high speeds should occur at smaller turning angles and finer sampling intervals. High speeds at large turning angles and sampling intervals are indicative of GPS errors, as in the case for animal E006. E002 and E003 do not show any clear issues. E002 has two points with high speeds and large turning angles that are worth checking, but plotting the adjacent telemetry points with `plot_tel()` indicates there are no issues. It is worth noting that all speed-based inferences are only appropriate if sampling frequency is fine enough to reconstruct at least an approximate movement path (Noonan *et al.*, 2019; Nathan *et al.*, 2022). The example data in Fig. 1 would be too coarse to make

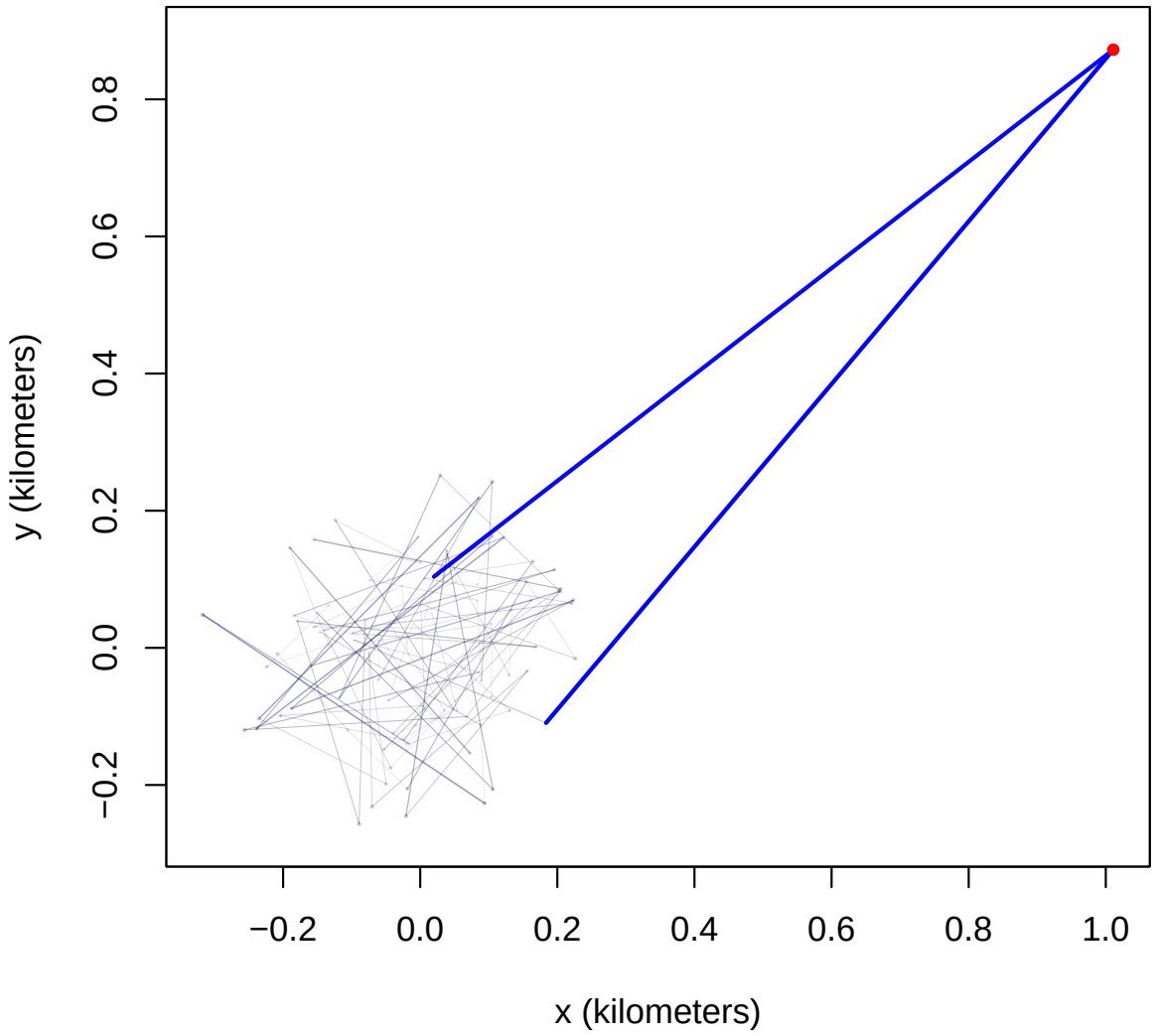
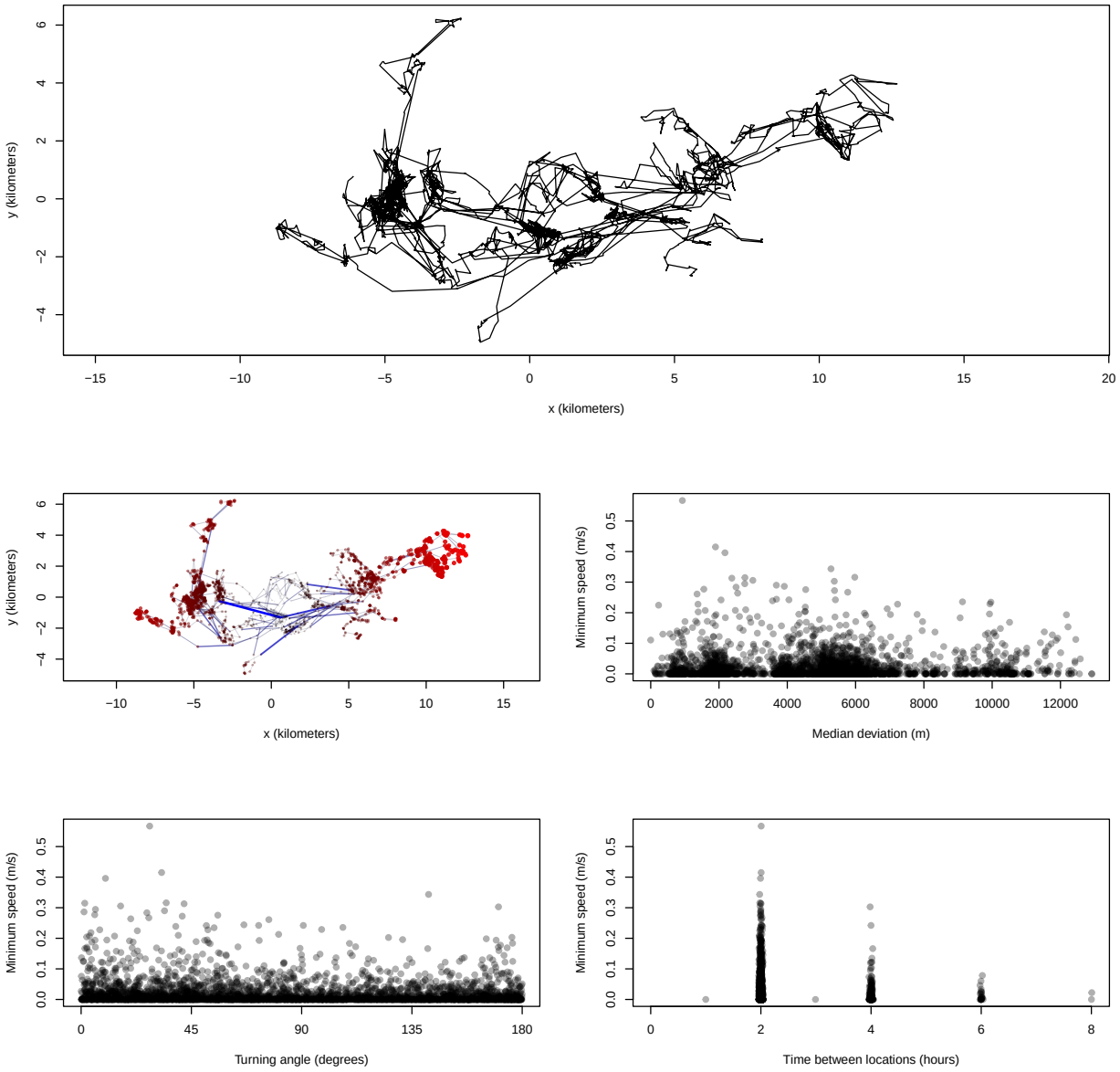


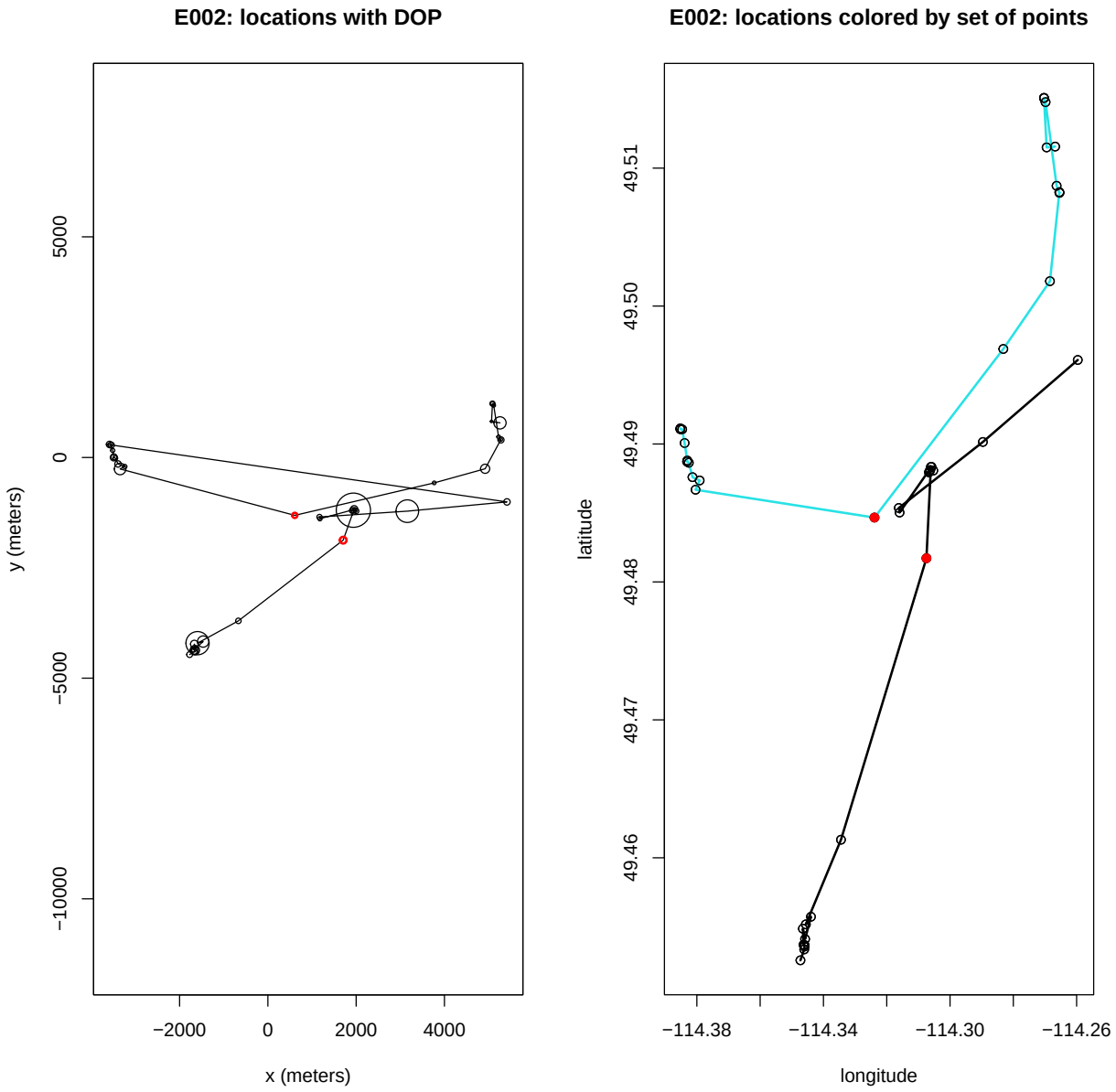
Figure A1: Example output from *extttctmm* :: *outlie*() using randomly generated bivariate Gaussian locations (dots) and a purpousefully erroneous location. Dots are larger and redder if they are further from the median location, and segments are thicker and bluer if the estimated straight-line displacement is high. Since the erroneous location is so far from the median location, both median deviation and minimum speed are quite high, so it is immediately visible. Note that the turning angle is large because the data implies a near-180-degree turn.

any inferences on minimum speed because there are no clear paths in the data: the animal seems to move around randomly (which is unsurprising, since the locations are random i.i.d. bivariate Gaussian draws).

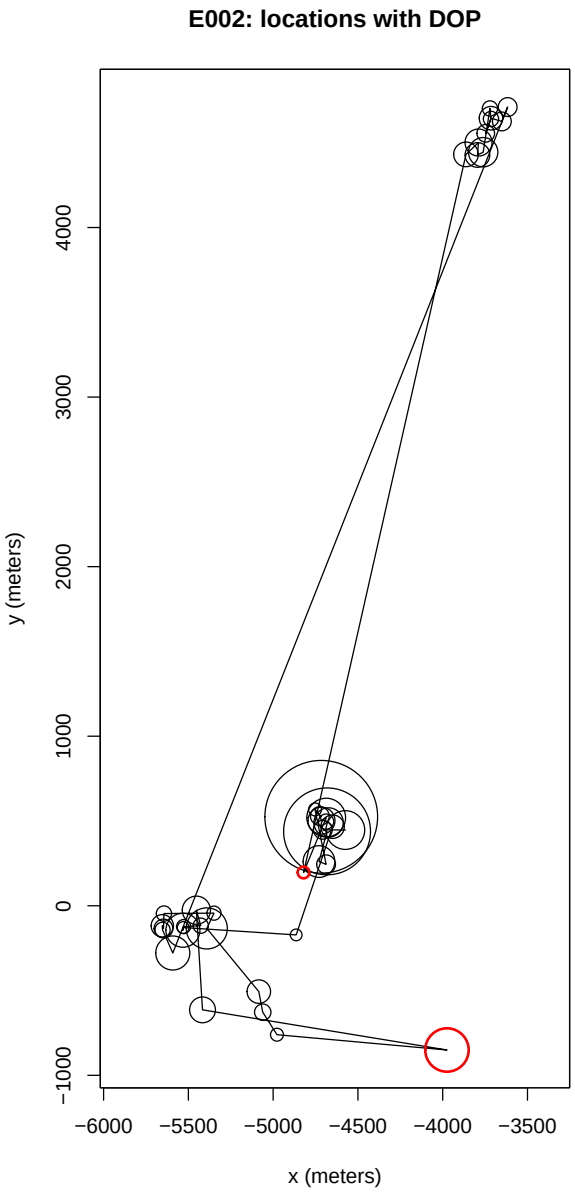
```
# E002
out <- check_animal('E002')
```

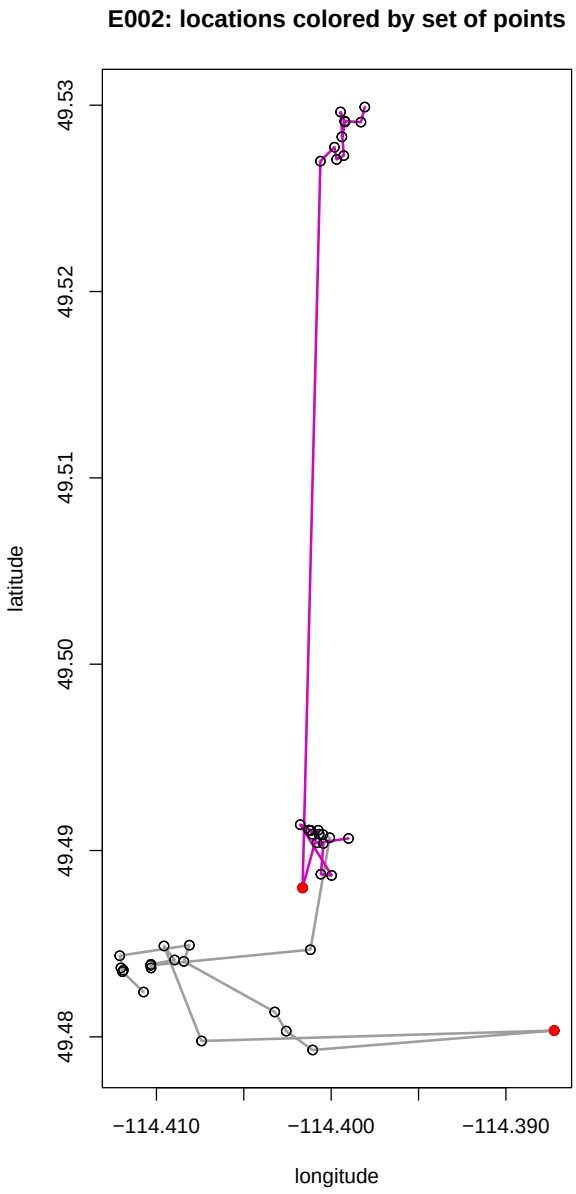



```
plot_adj('E002', max_speed = 0.4)
```

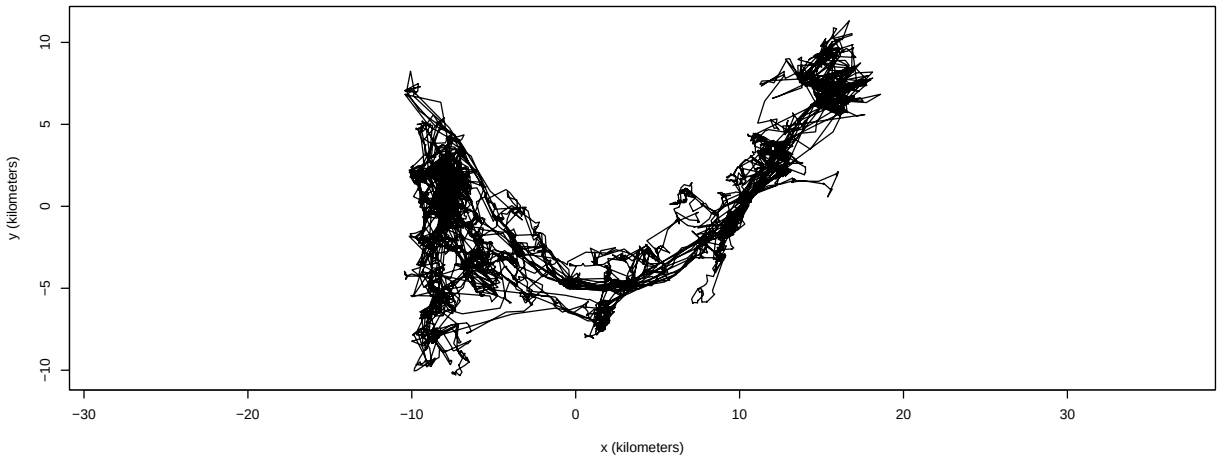


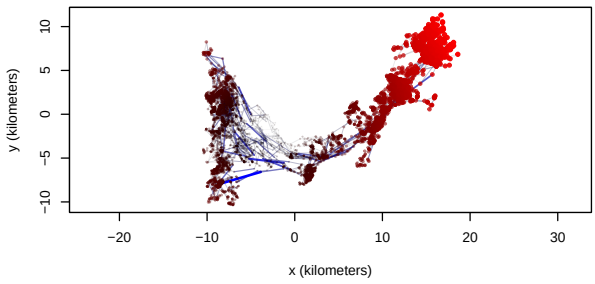

```
plot_adj('E002', max_angle = 170, max_speed = 0.2)
```

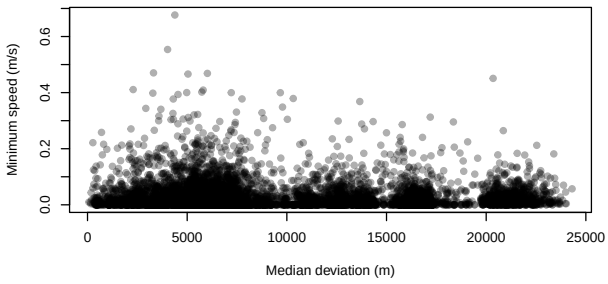


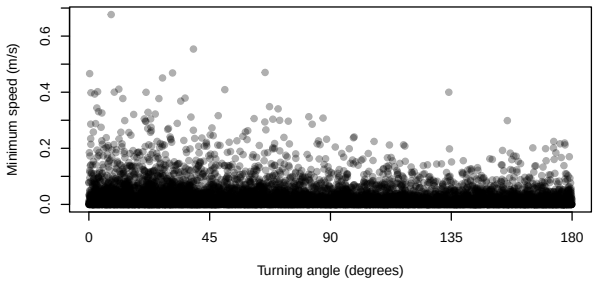


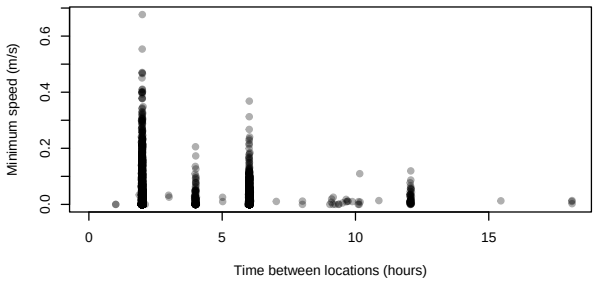

```
# E003
out <- check_animal('E003')
```



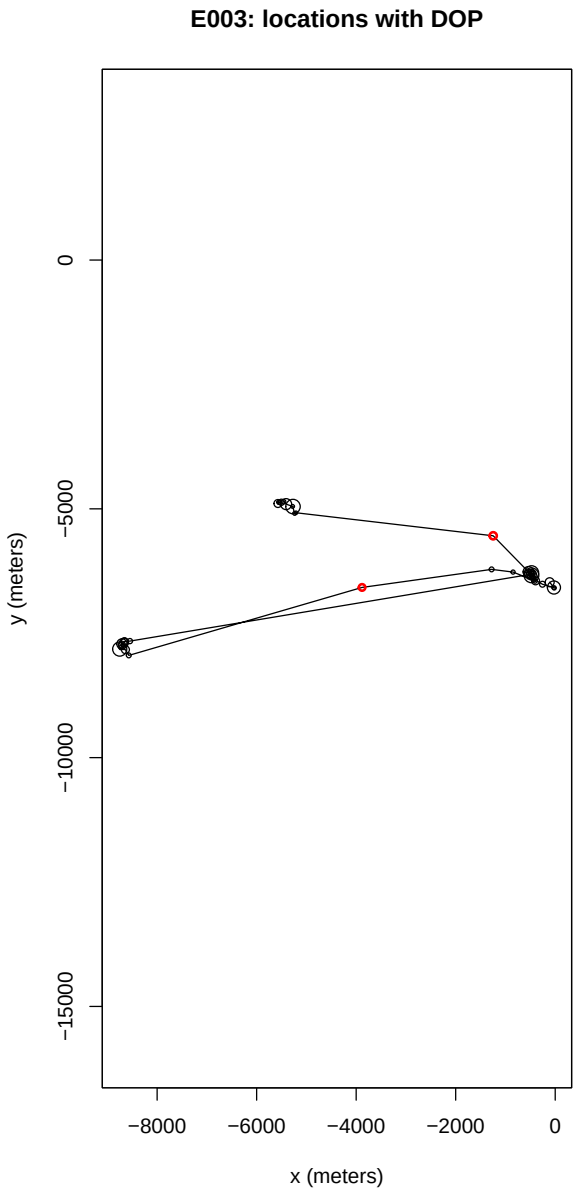


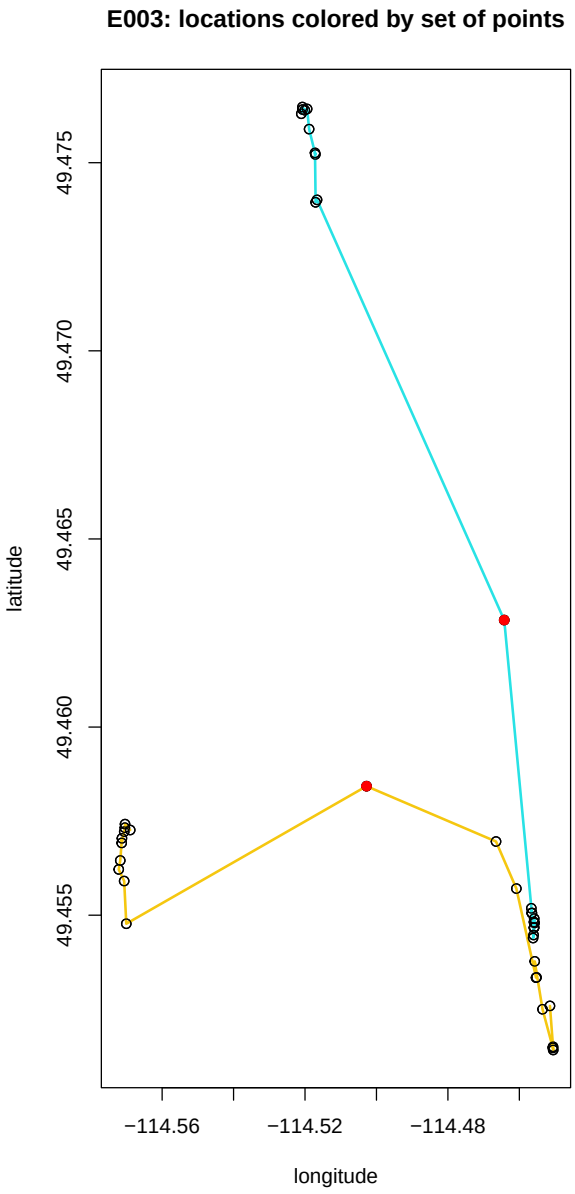




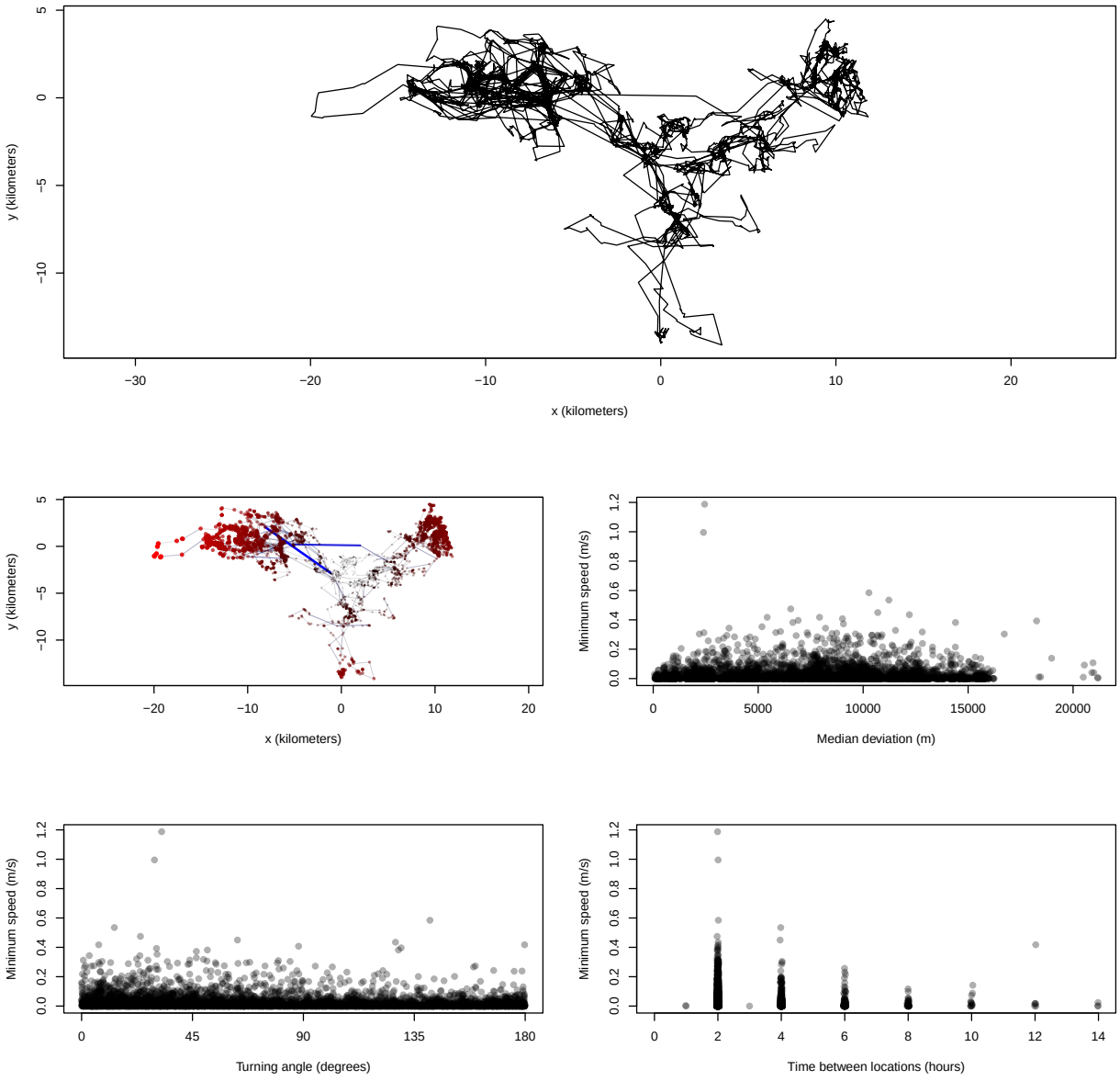



```
plot_adj('E003', max_speed = 0.5)
```

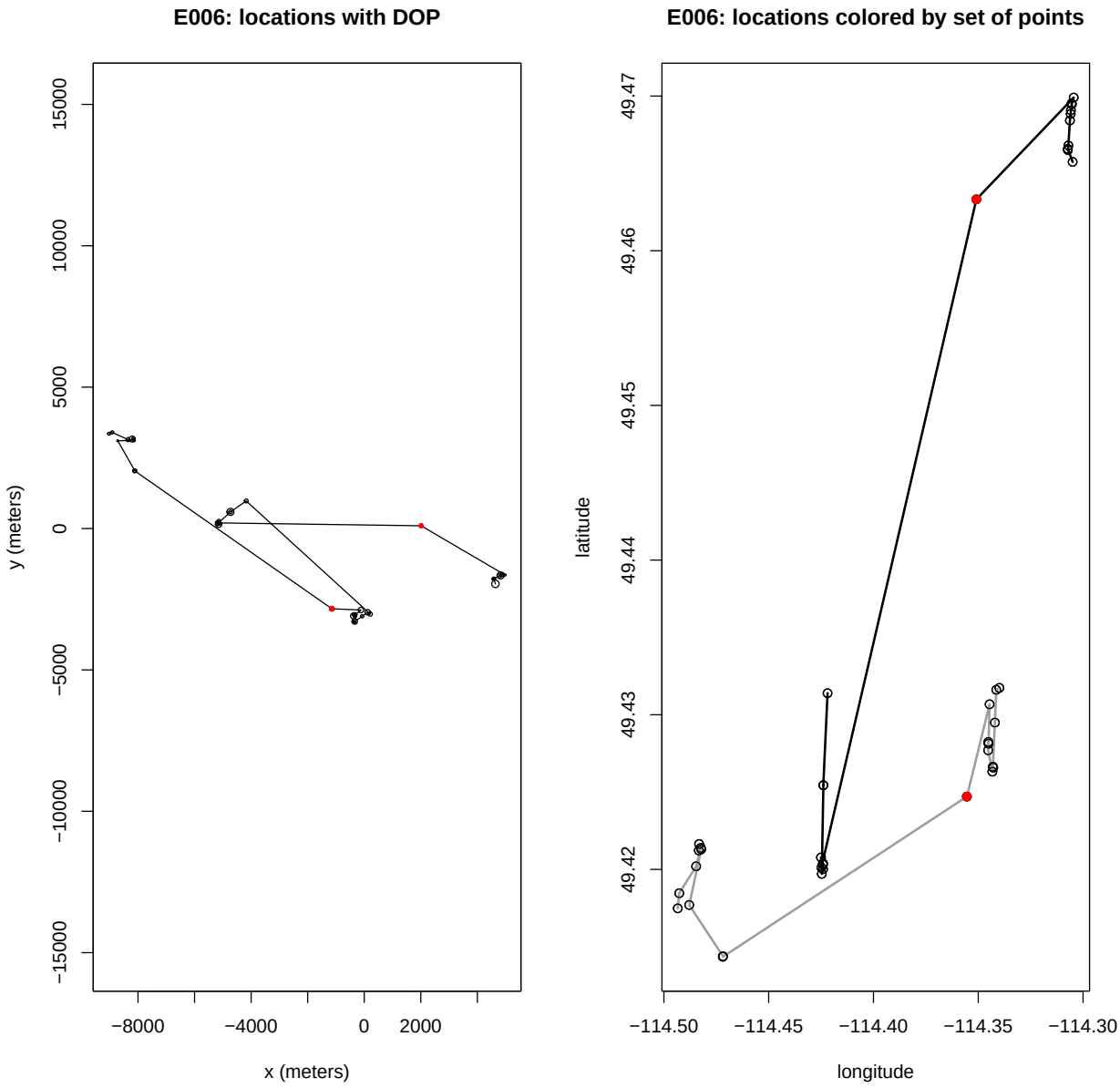




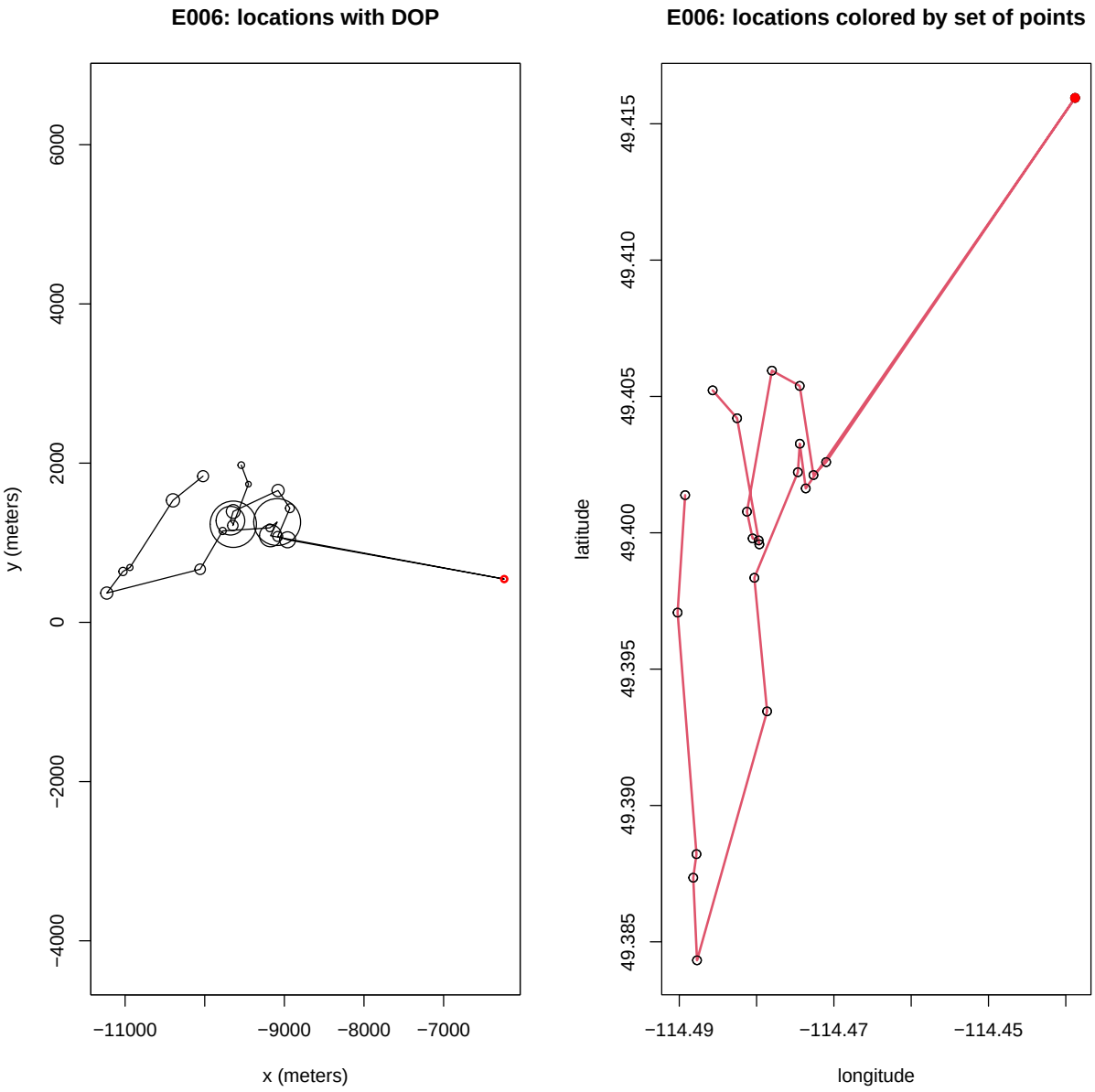

```
# E006
out <- check_animal('E006')
```




```
plot_adj('E006', max_speed = 0.8) # ok
```




```
plot_adj('E006', max_speed = 0.4, max_angle = 170) # outlier
```



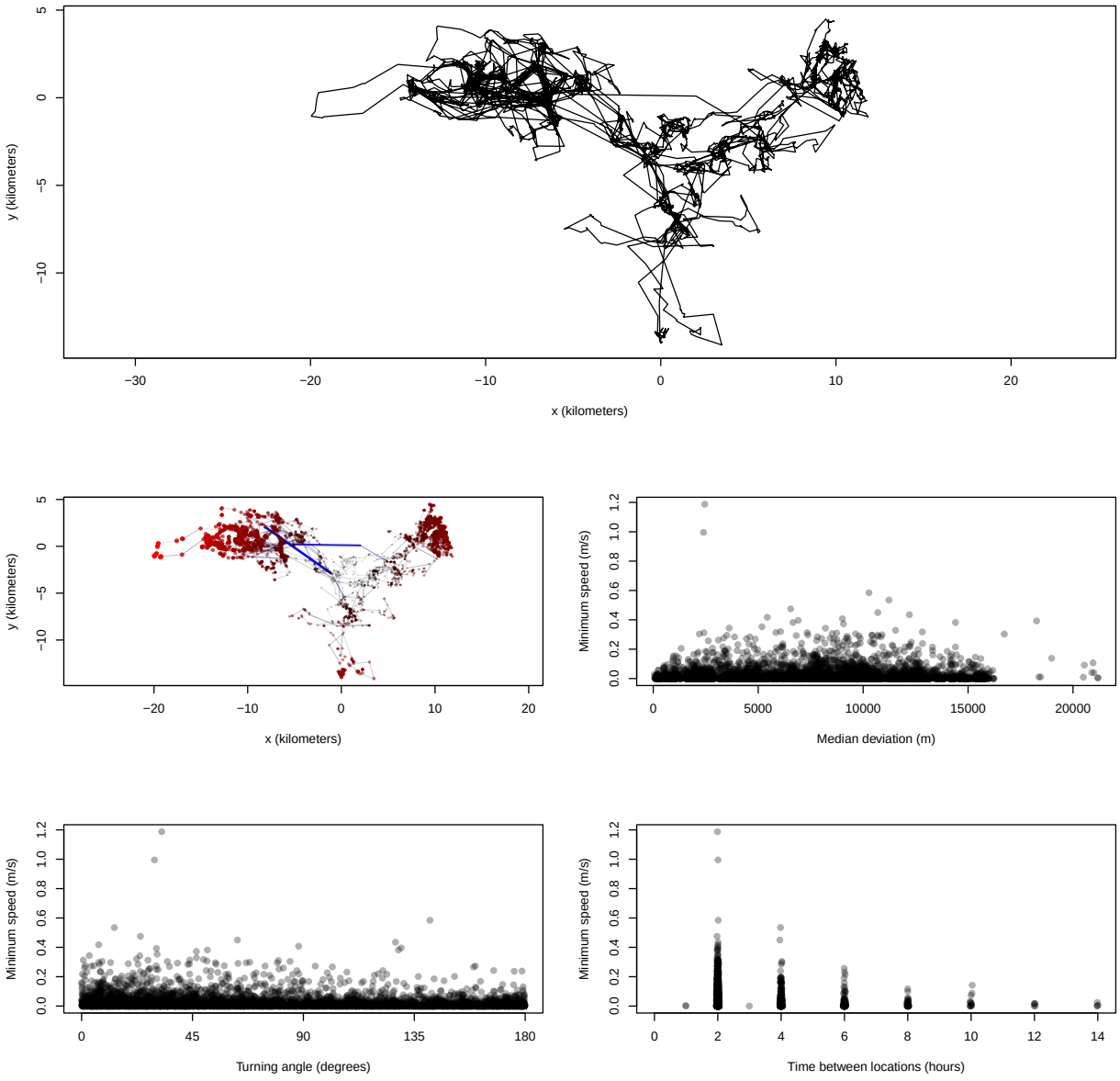

```
plot_adj('E006', max_speed = 0.4, max_angle = 170, map = FALSE) # see river
# dt is much larger: collar struggled to receive signal with tree cover
'hours' %#% out[which(out$speed > 0.4 & out$angle > 170) + (-4:4), 'dt']
```

```
## [1] 10.007222  2.013056  3.981111  1.994444 12.022778  1.985556  2.015000
## [8]  1.970556  4.027222
```

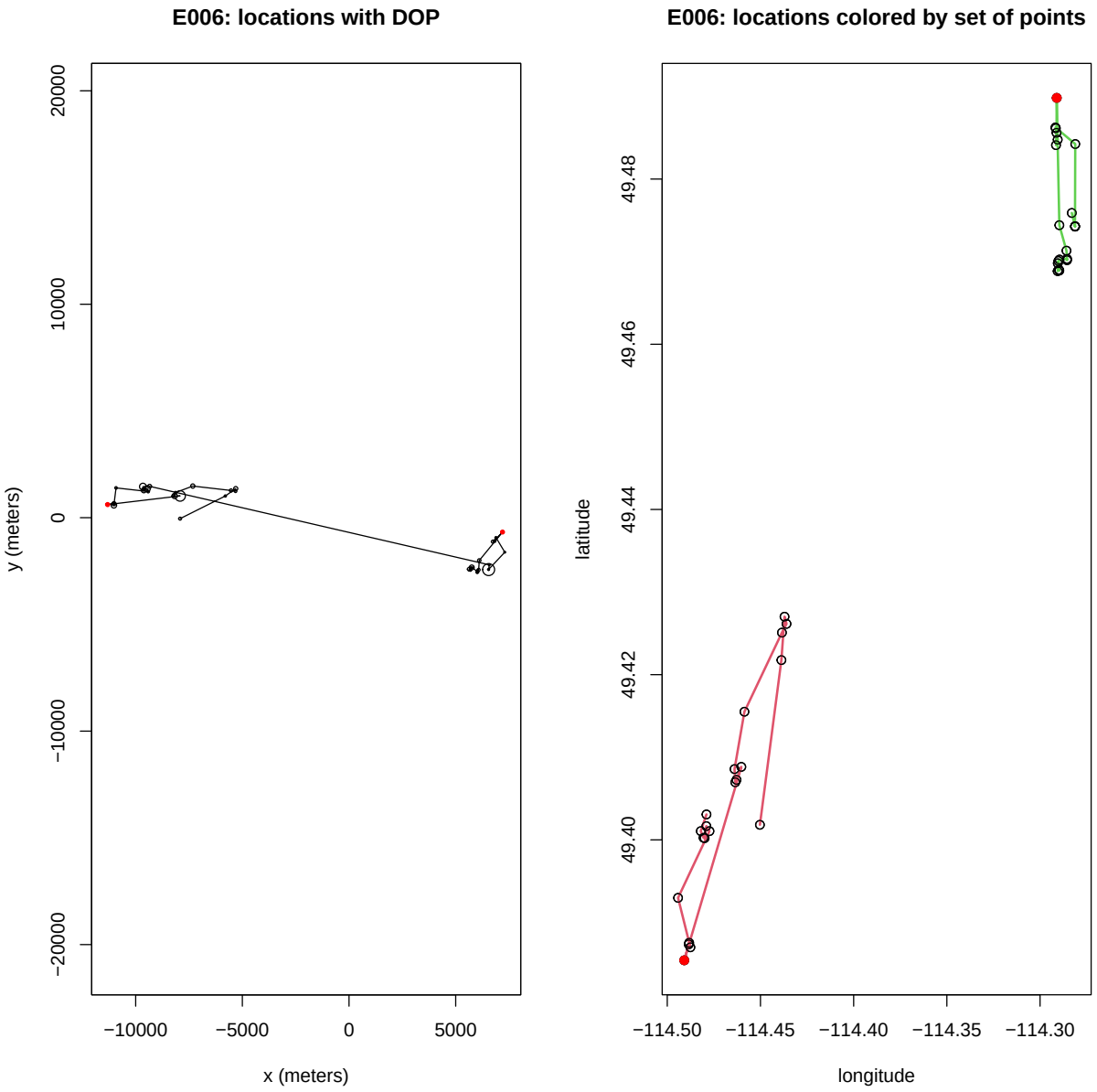
```
'hours' %#% c(43282, 7254) # convert seconds to hours
```

```
## [1] 12.02278  2.01500
```

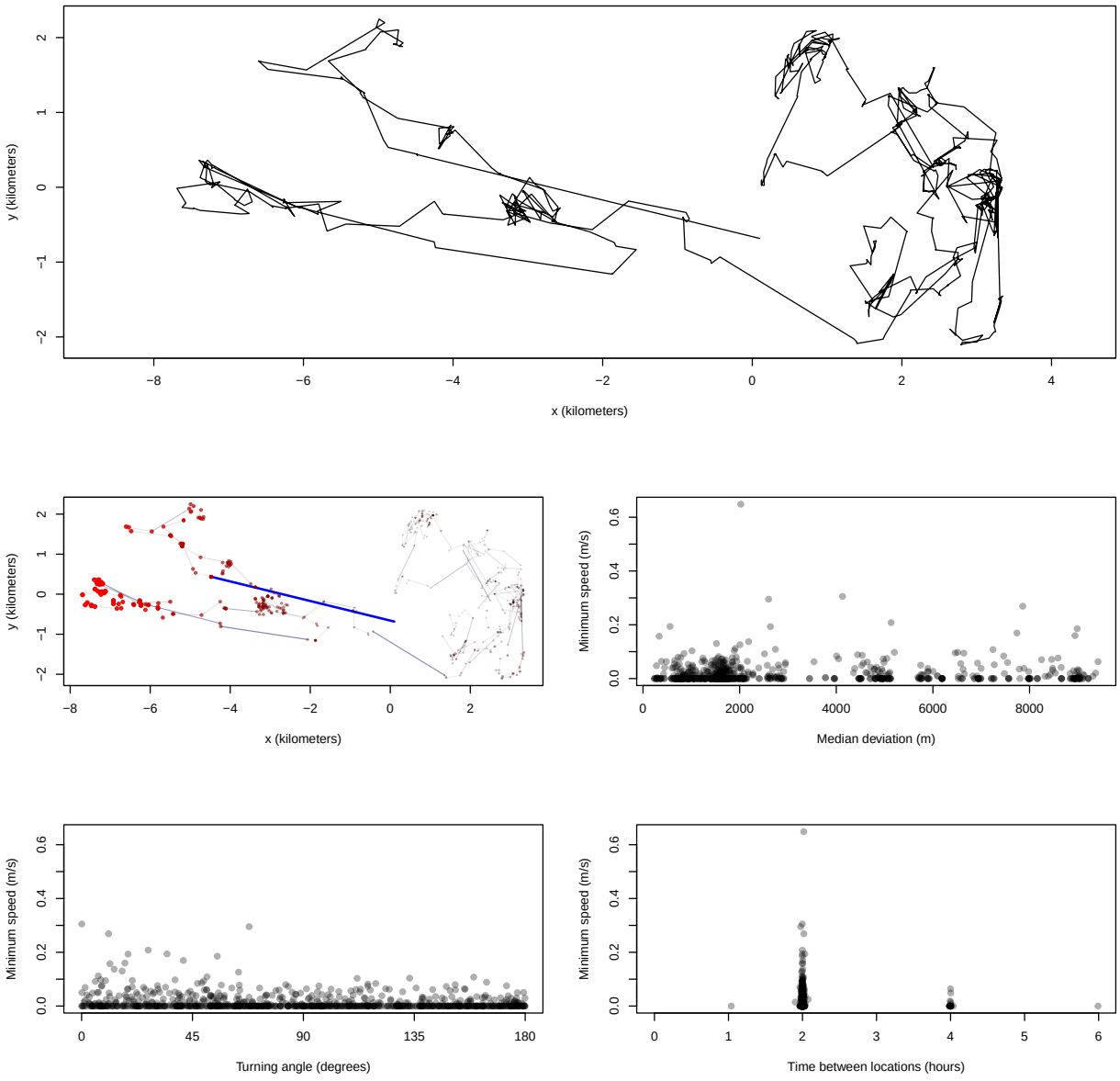
```
flag_outlier('E006', max_speed = 0.4, max_angle = 170, value = 1)
out <- check_animal('E006')
```



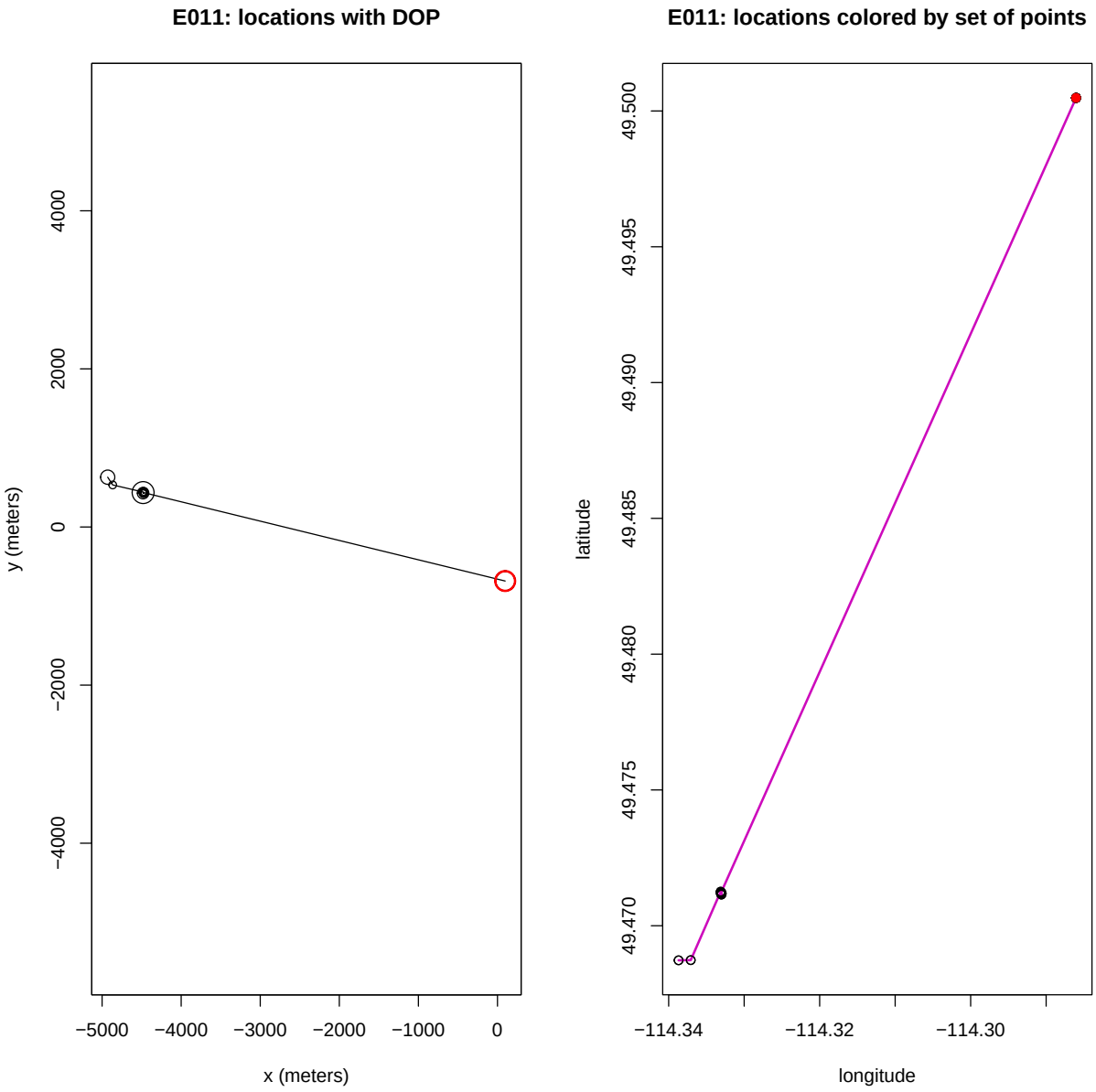

```
plot_adj('E006', max_speed = 0.2, max_angle = 170) # ok
```




```
# E011: last point is outlier (keeping previous points bc many 0 speeds)
out <- check_animal('E011')
```



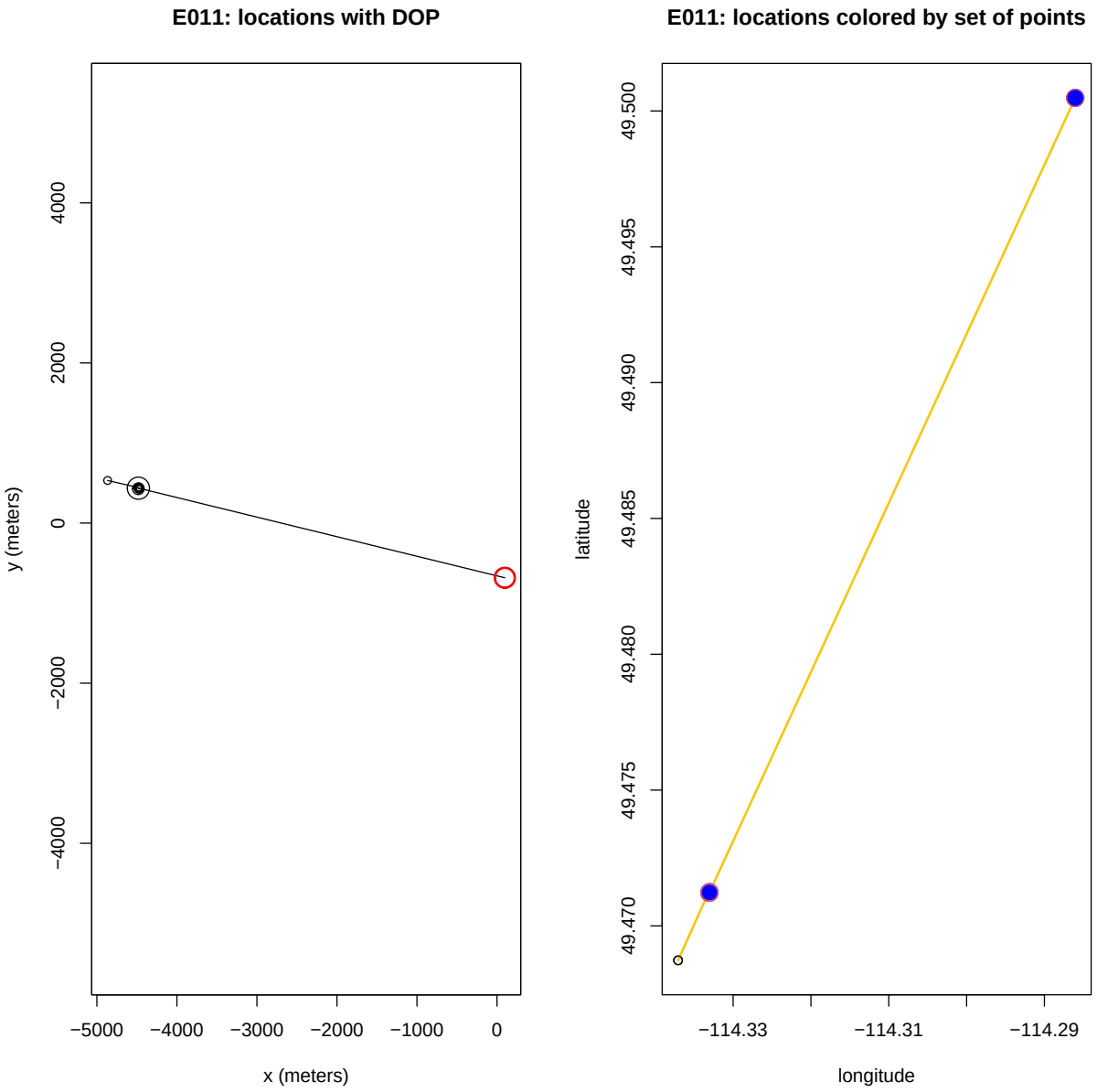

```
plot_adj('E011', max_speed = 0.4, n_adj = 10)
```



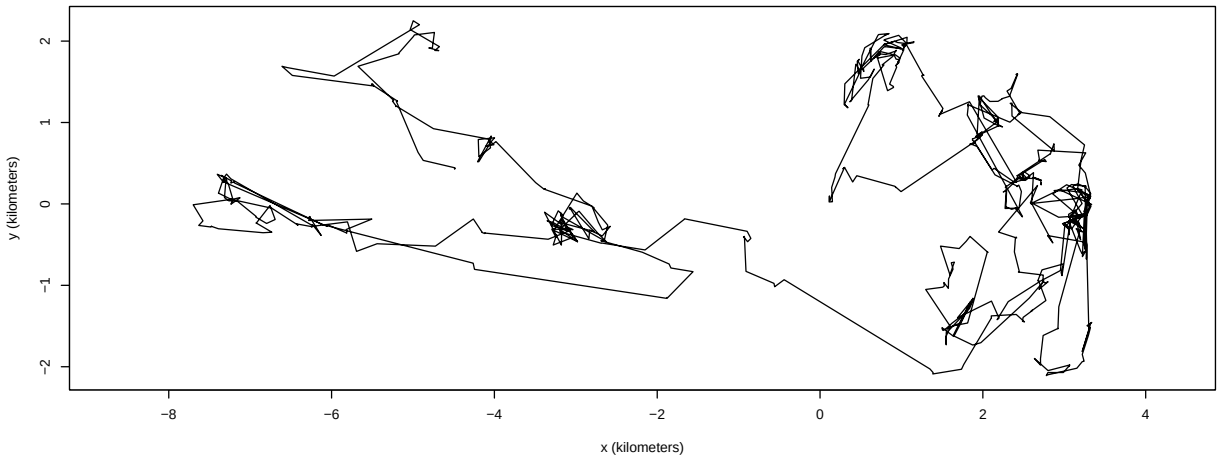

```
plot_adj('E011', max_speed = 0.4, n_adj = 10, map = FALSE)
plot_adj('E011', max_speed = 0.4, n_adj = 9, reset_layout = FALSE)
which(out$speed > 0.4) == nrow(out)
```

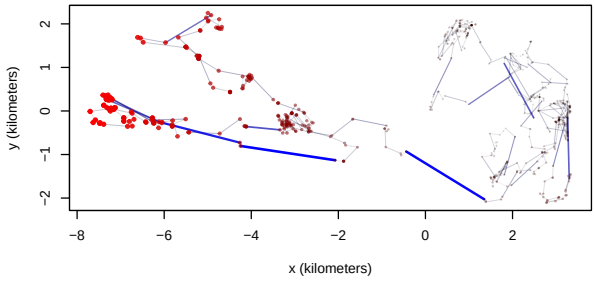
```
## [1] TRUE
```

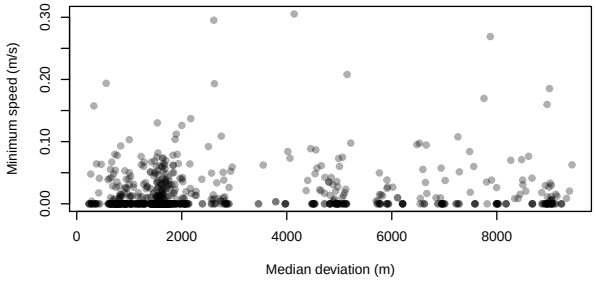
```
i_tel <- which(d$animal == 'E011')
i <- (nrow(d$tel[[i_tel]]) - 7):nrow(d$tel[[i_tel]]) # drop last 8 points
points(location.lat ~ location.long, d$tel[[i_tel]][i, ], col = 'blue',
       cex = 2, pch = 19)
d$tel[[i_tel]][i, 'outlier'] <- 1
points(location.lat ~ location.long, d$tel[[i_tel]][i, ],
       col = as.numeric(d$tel[[i_tel]]$outlier[i]) + 1, cex = 2)
```

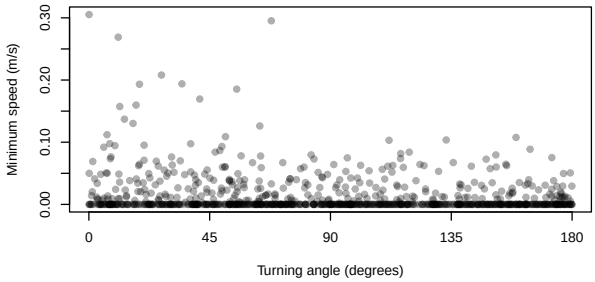


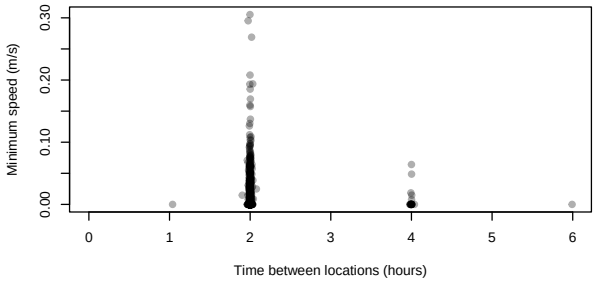

```
layout(1)

out <- check_animal('E011')
```



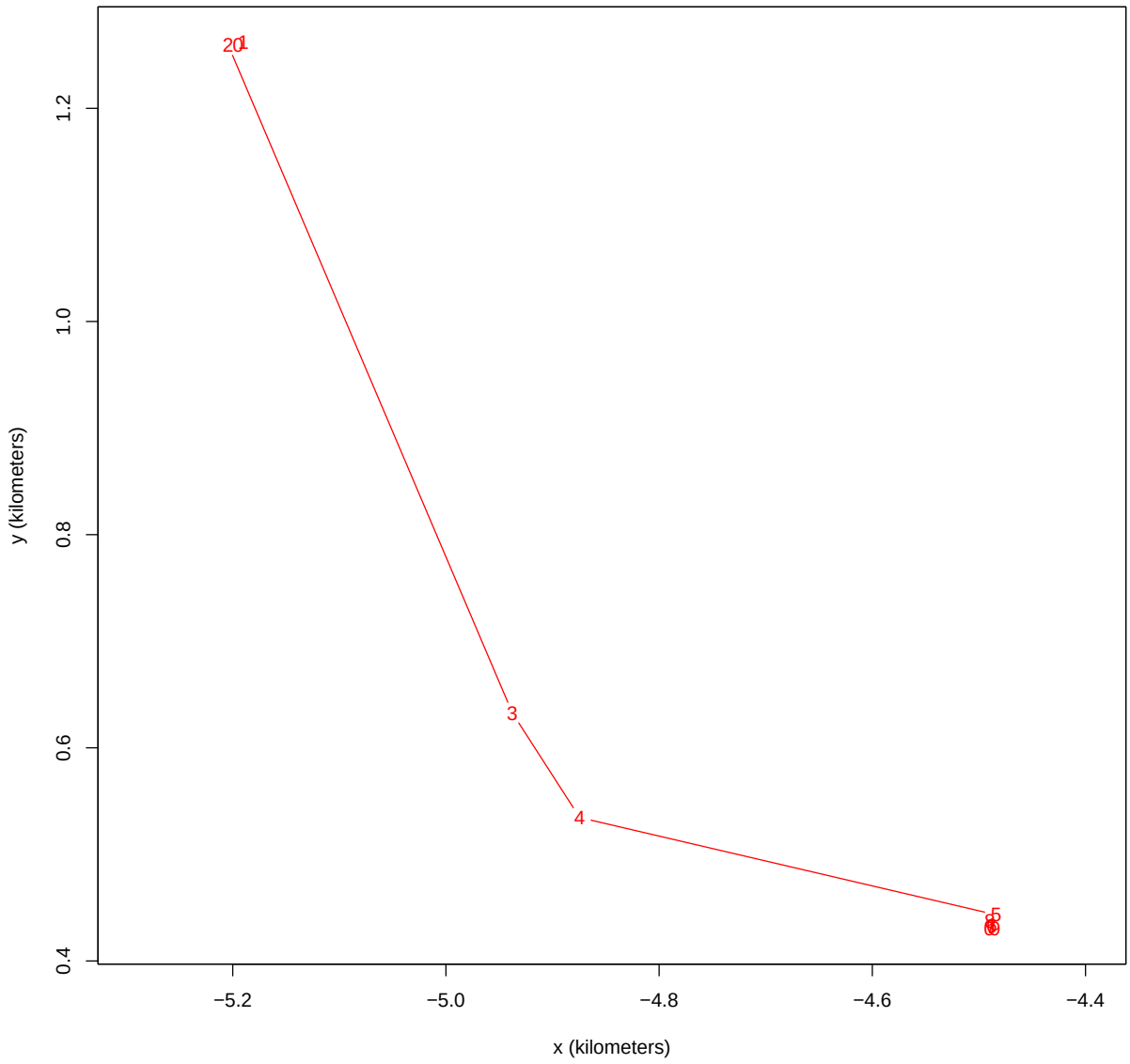




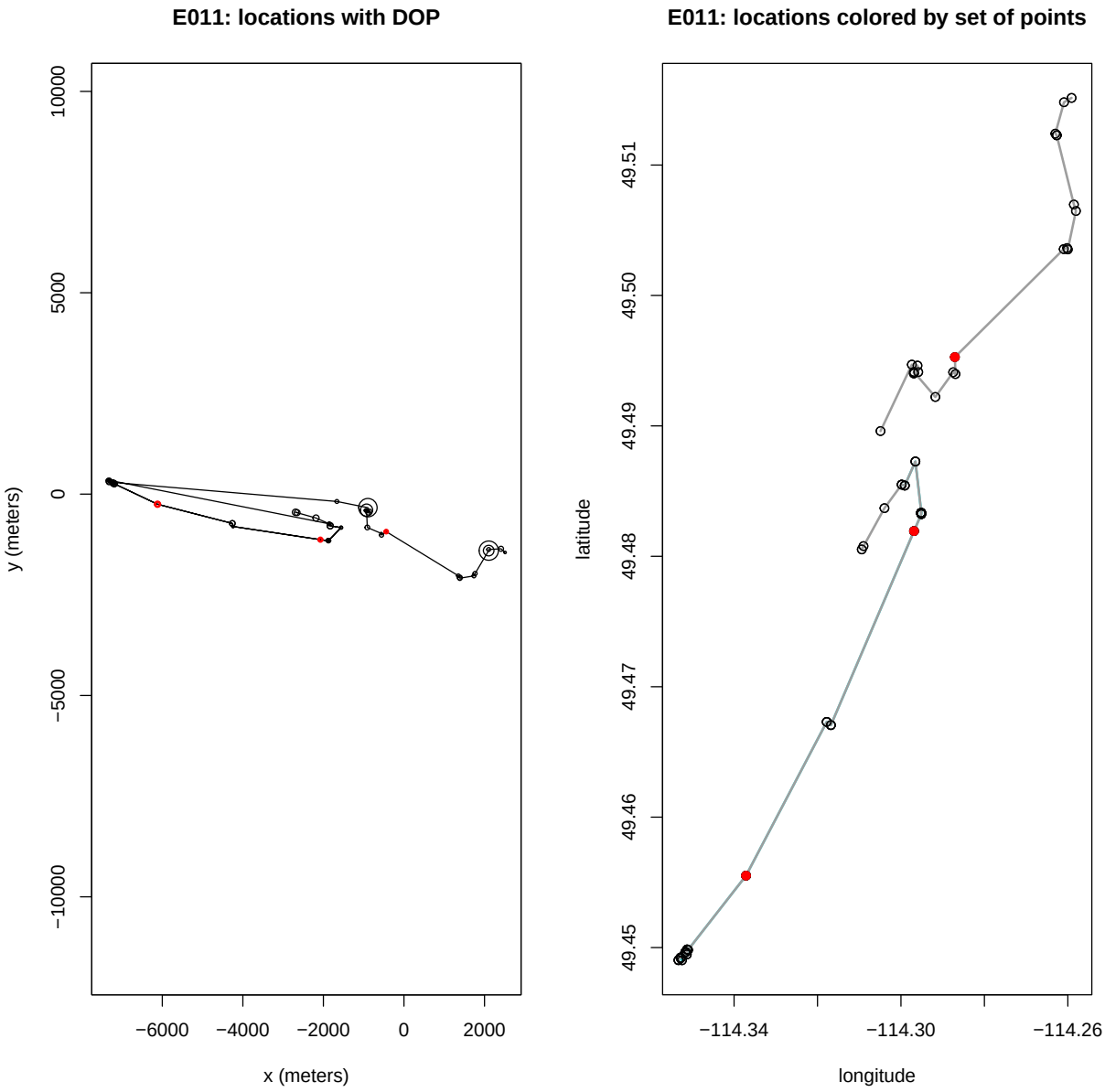




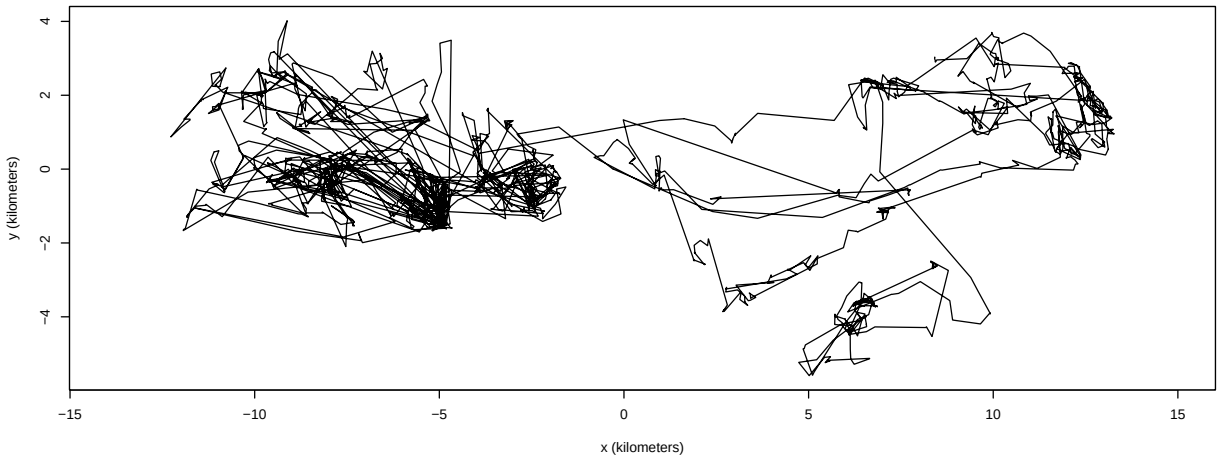

```
tel <- as.telemetry(d$tel[[i_tel]], mark.rm = TRUE)
plot(tel[nrow(tel) + -10:0, ], error = FALSE, type = 'b',
     pch = as.character(0:9))
```

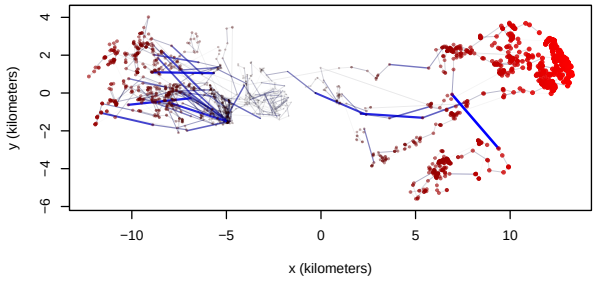


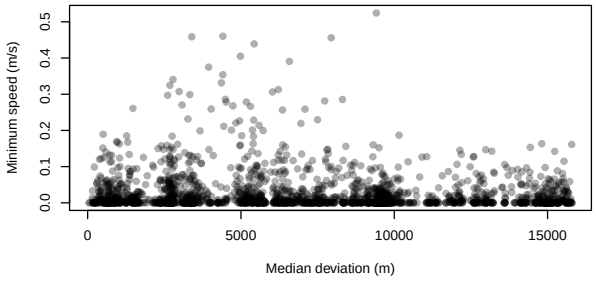

```
plot_adj('E011', max_speed = 0.25)
```

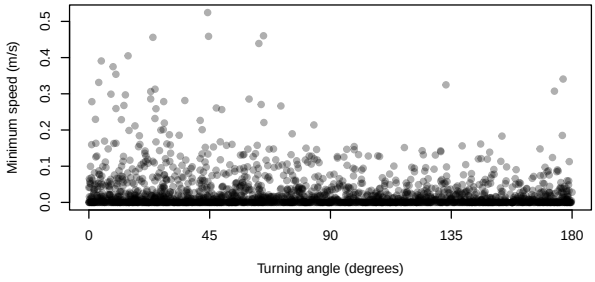


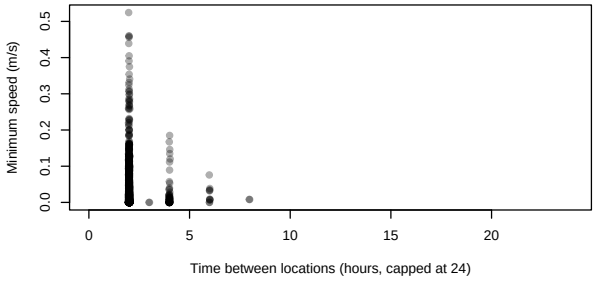

```
# E013
out <- check_animal('E013')
```



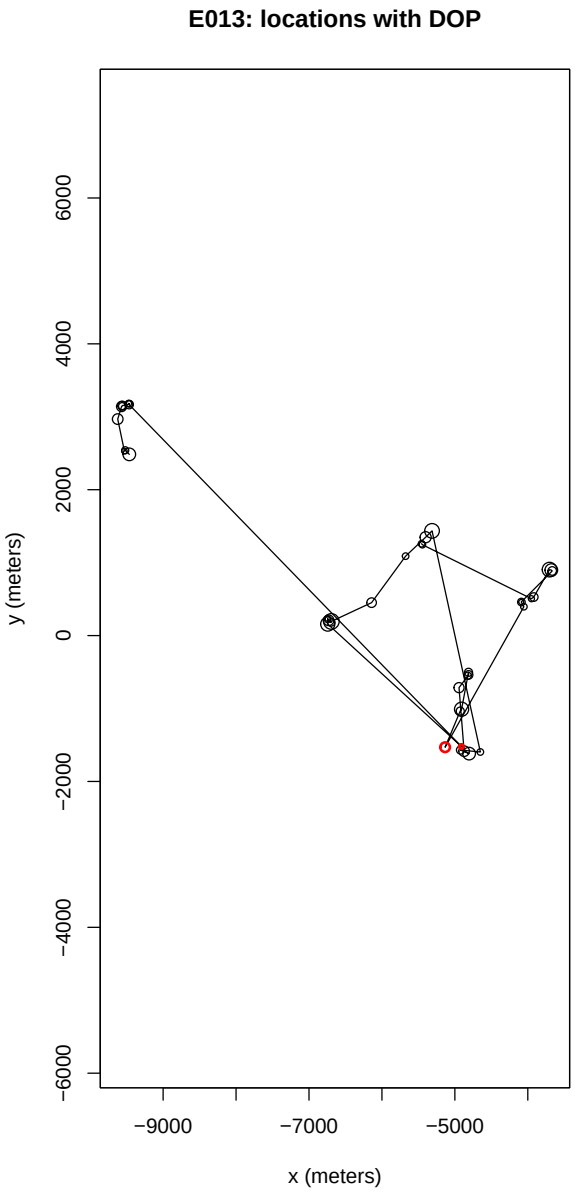


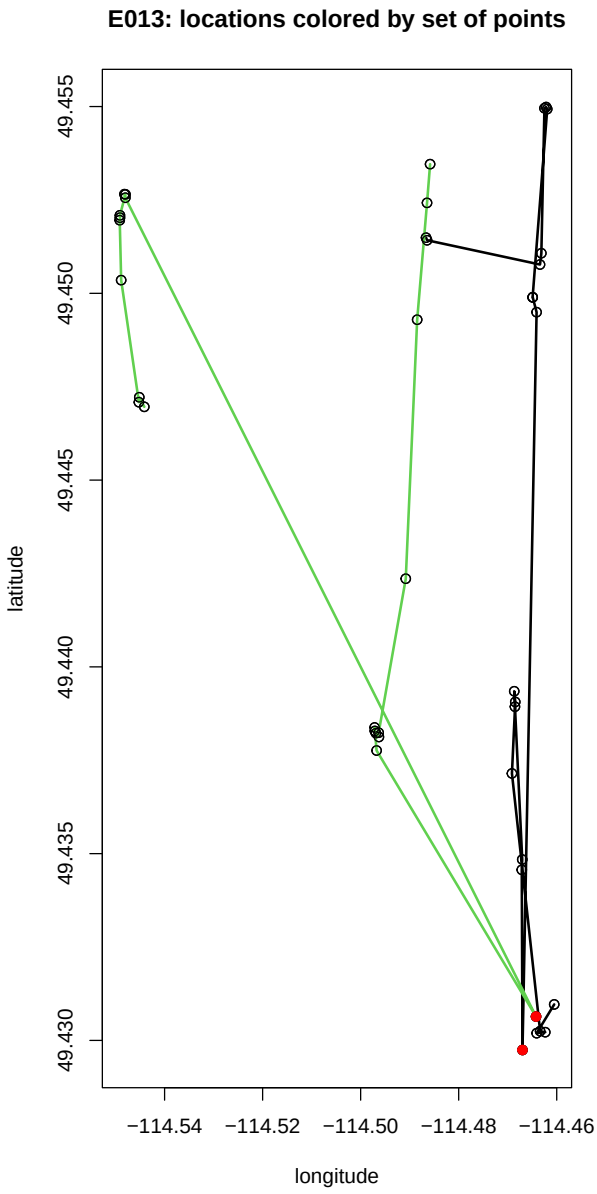




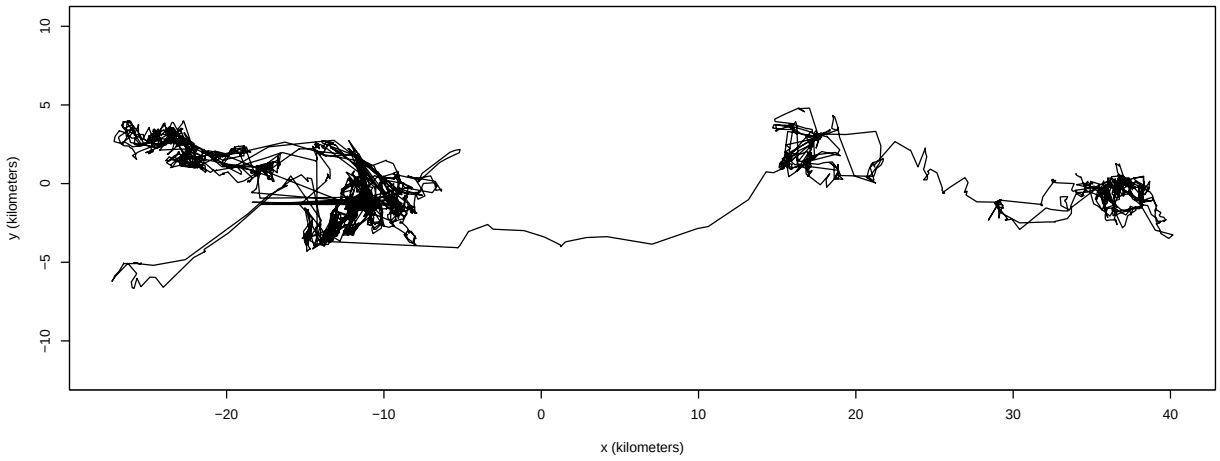


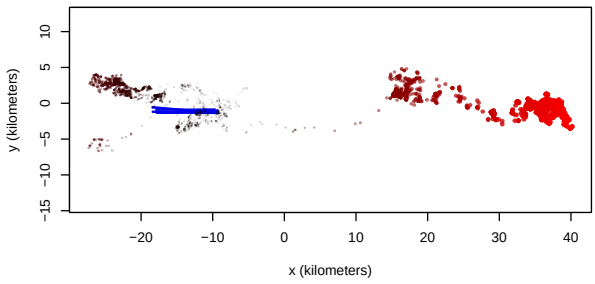

```
plot_adj('E013', max_speed = 0.2, max_angle = 170) # escaped something?
```

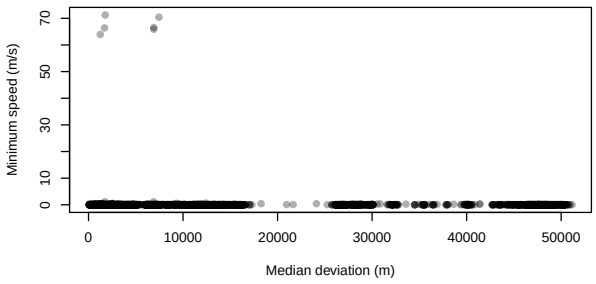


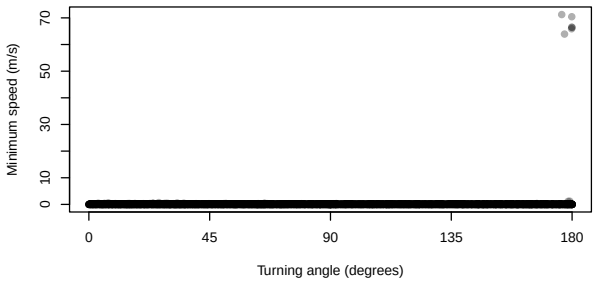


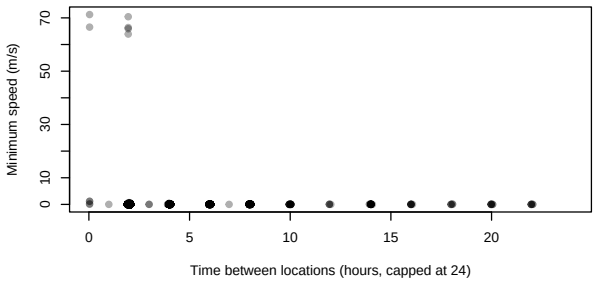

```
# E015 has multiple GPS malfunctions
out <- check_animal('E015')
```





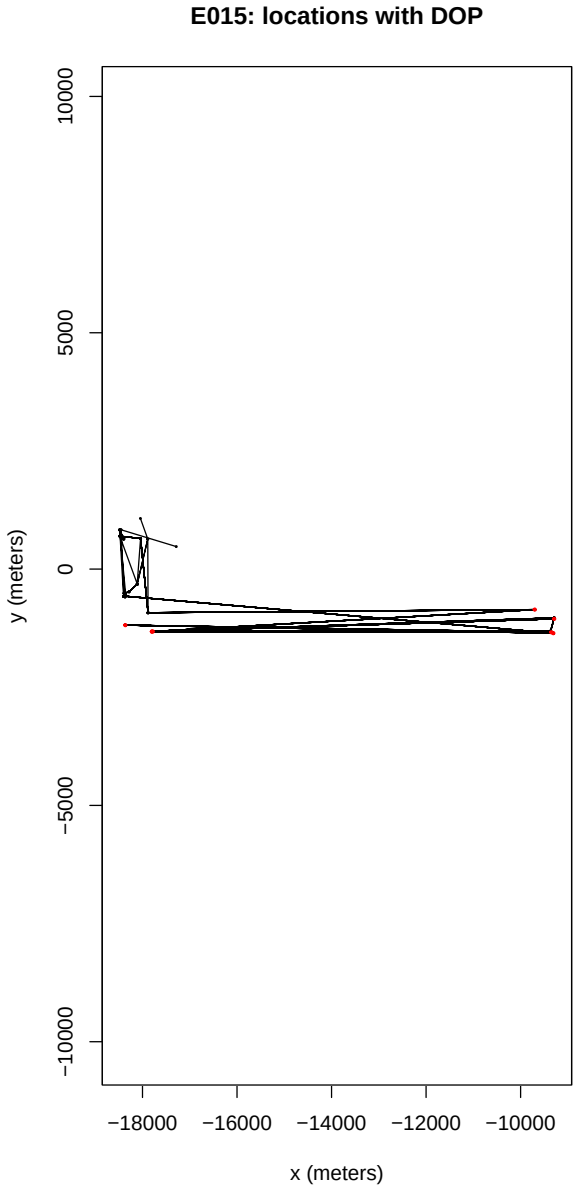


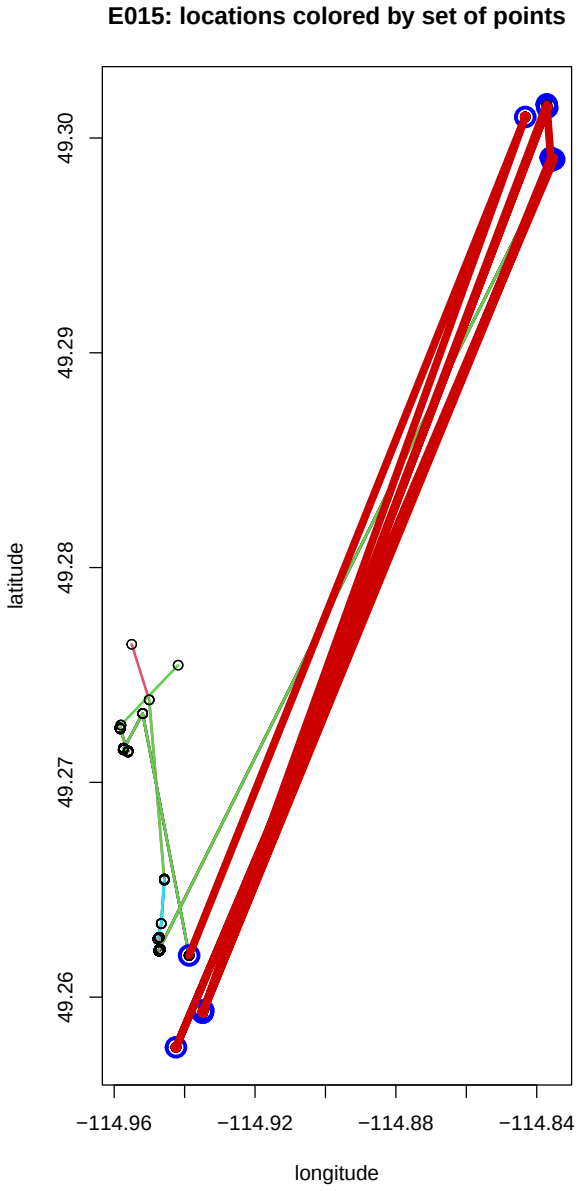




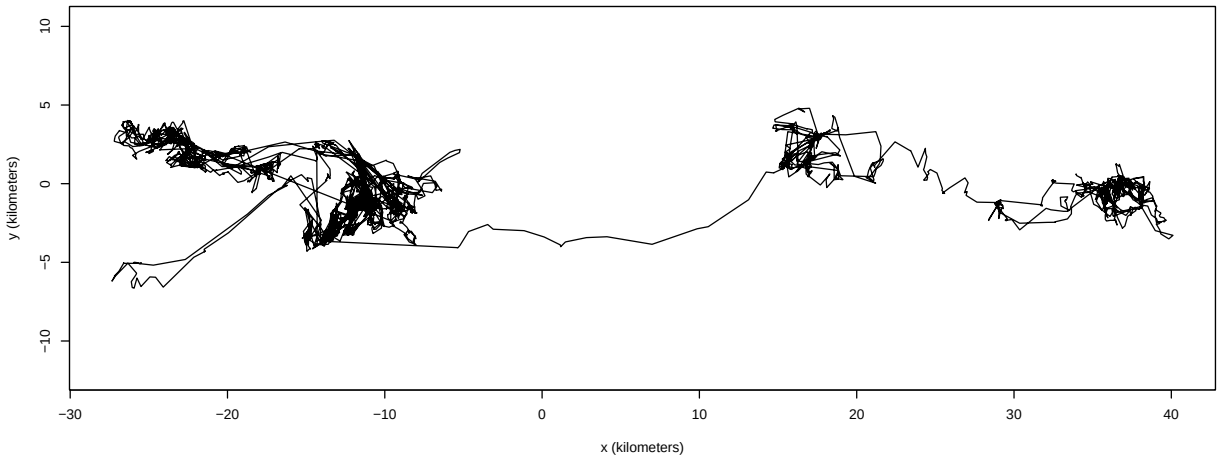

```
plot_adj('E015', max_speed = 1, reset_layout = FALSE)
# can't filter easily because the outliers don't have the max velocity
i <- d %>%
  pull(tel) %>%
  nth(which(d$animal == 'E015')) %>%
  mutate(i = 1:n()) %>%
  slice(which(timestamp ==
                as.POSIXct(out[which(out$speed > 0.6)[2], 't'])) +
          0:1) %>%
  pull(i) %>%
  first() + 0:10
points(location.lat ~ location.long, cex = 2, lwd = 3, col = 'blue',
```

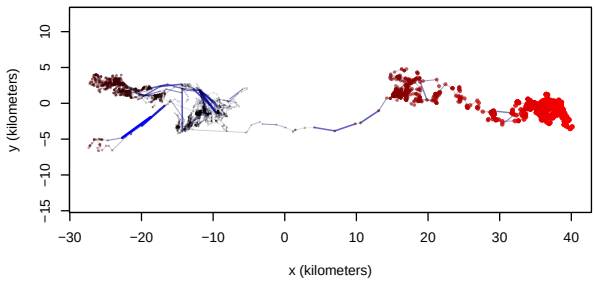
```
       filter(d, animal == 'E015')$tel[[1]][i, ])
lines(location.lat ~ location.long,
      filter(d, animal == 'E015')$tel[[1]][i, ], lwd = 6, col = 'red3')
```

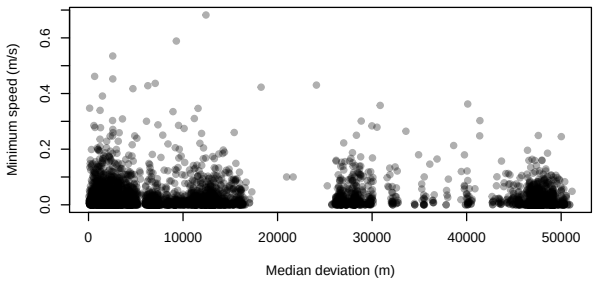


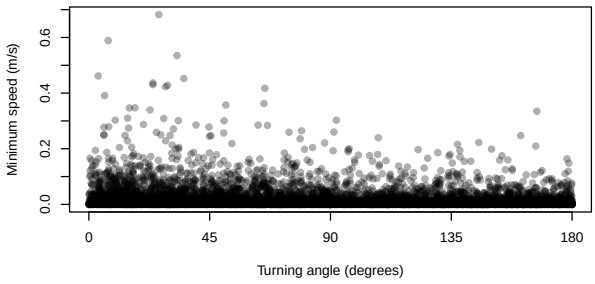


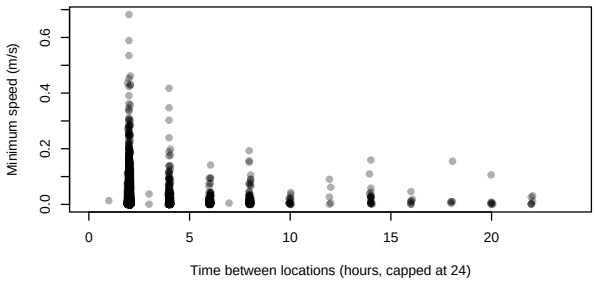

```
d$tel[[which(d$animal == 'E015')]][i, 'outlier'] <- 1
out <- check_animal('E015')
```



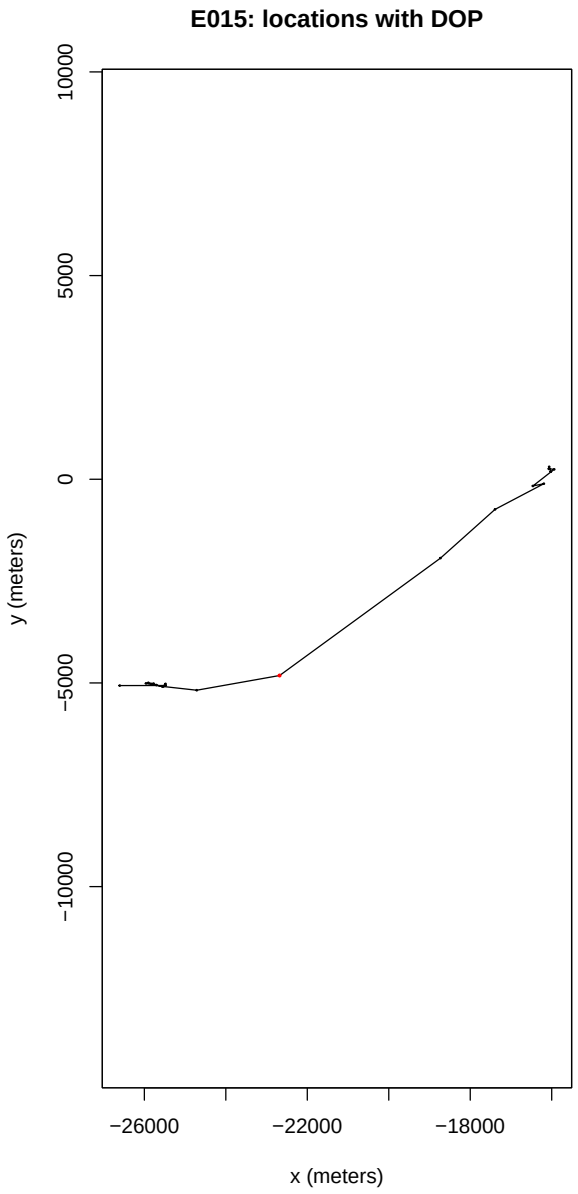


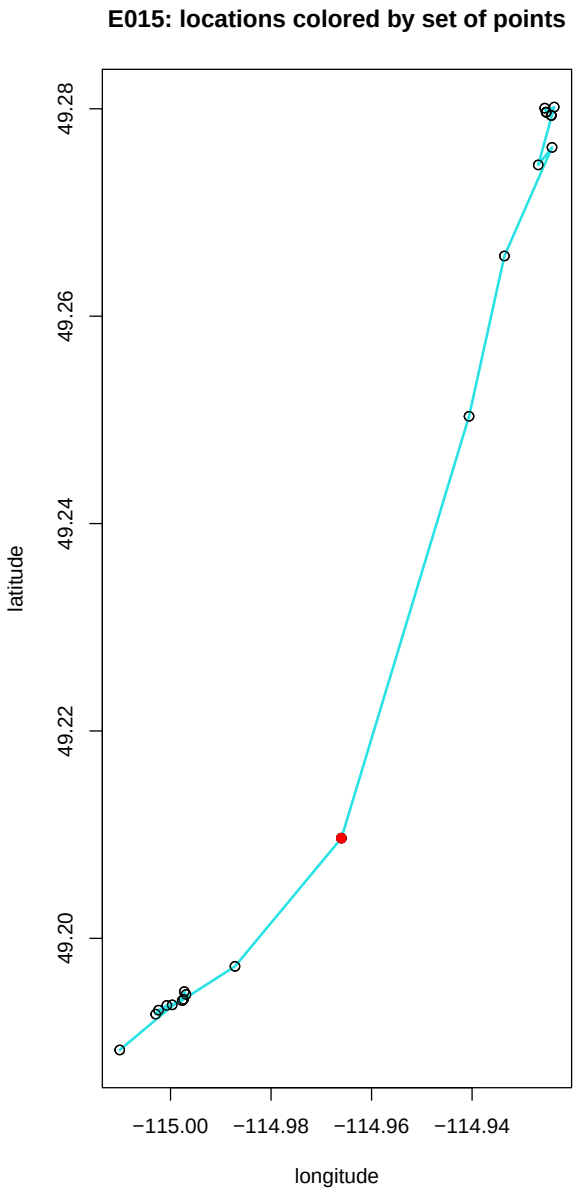




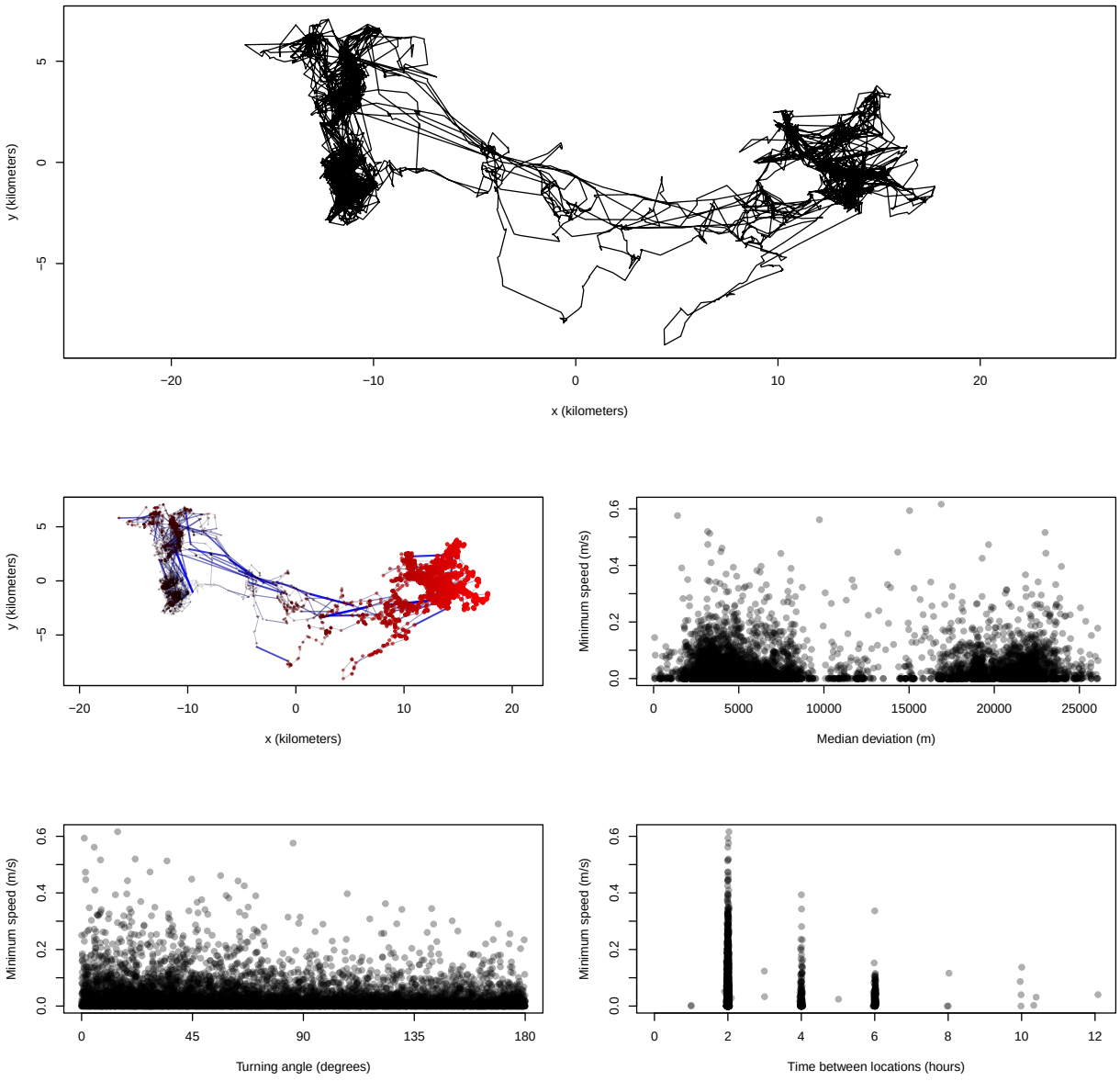



```
plot_adj('E015', max_speed = 0.60) # track looks ok now
```

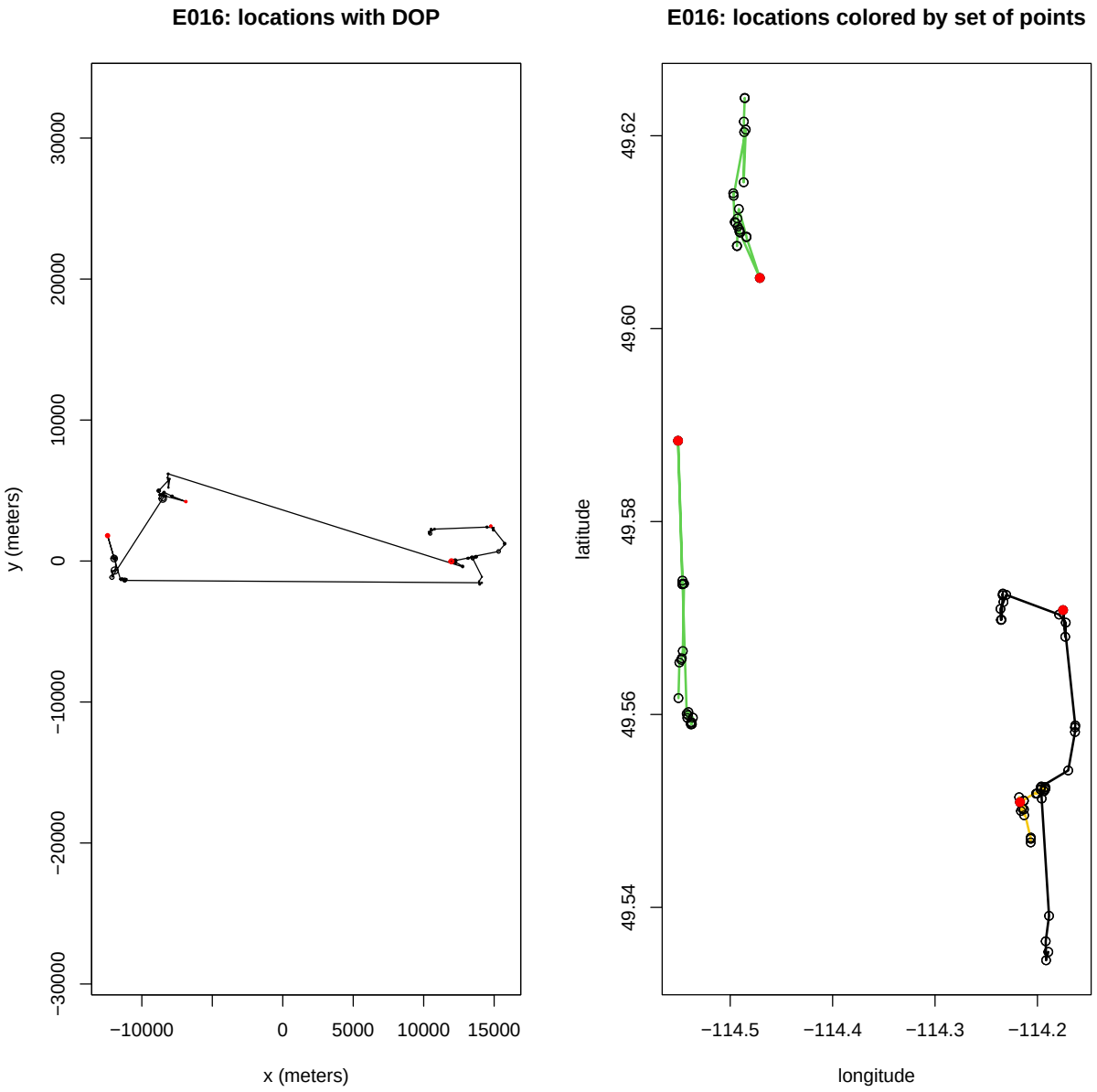




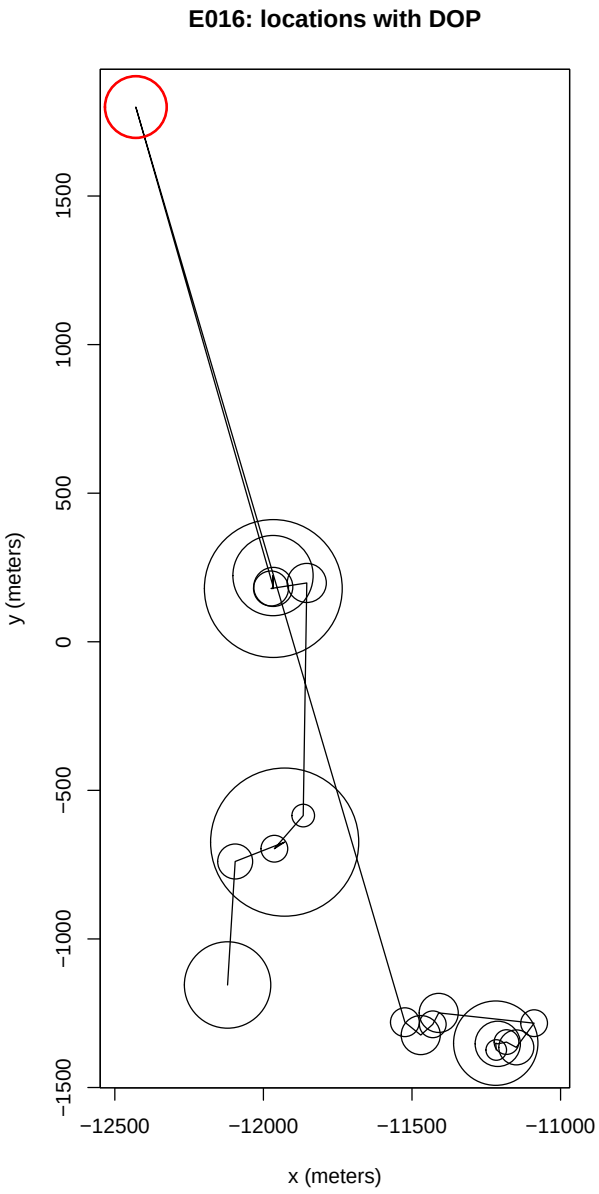

```
# E016
out <- check_animal('E016')
```

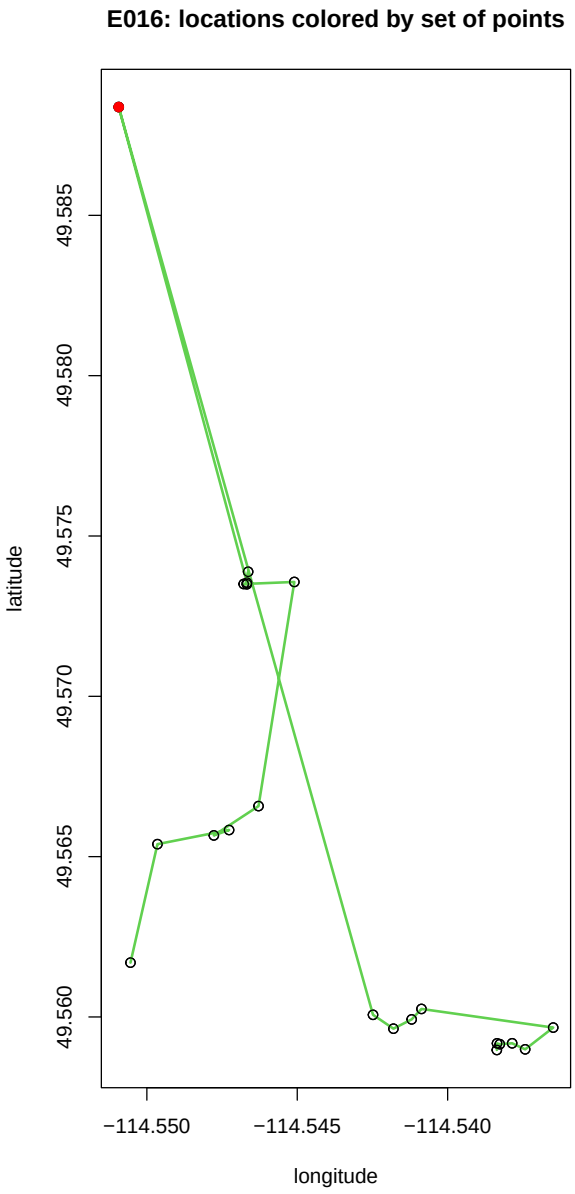



```
plot_adj('E016', max_speed = 0.2, max_angle = 170) # one outlier
```

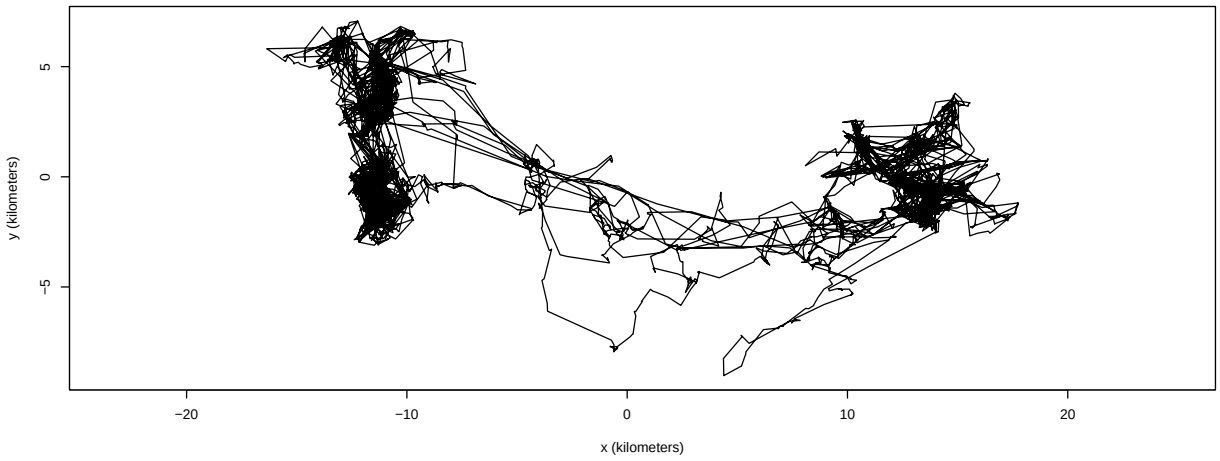


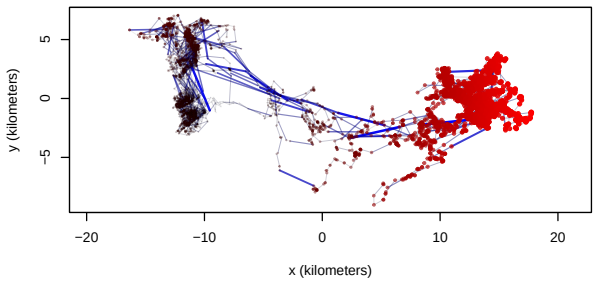

```
plot_adj('E016', max_speed = 0.2, max_angle = 179) # use higher upper bound
```

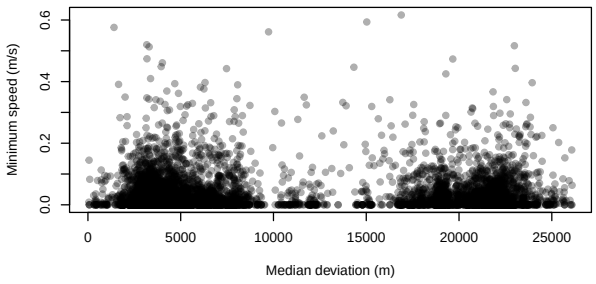


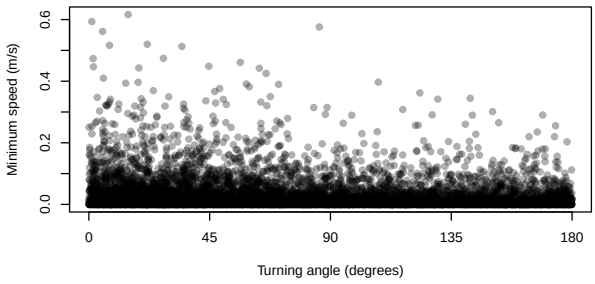


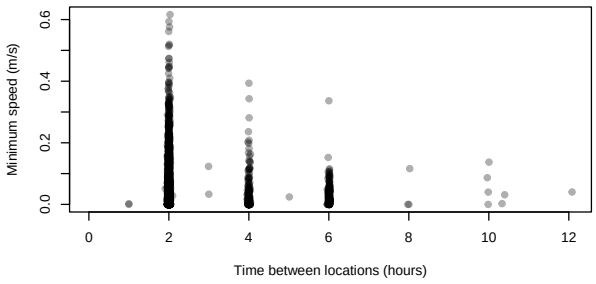

```
flag_outlier('E016', max_speed = 0.2, max_angle = 179, value = 1)
out <- check_animal('E016')
```



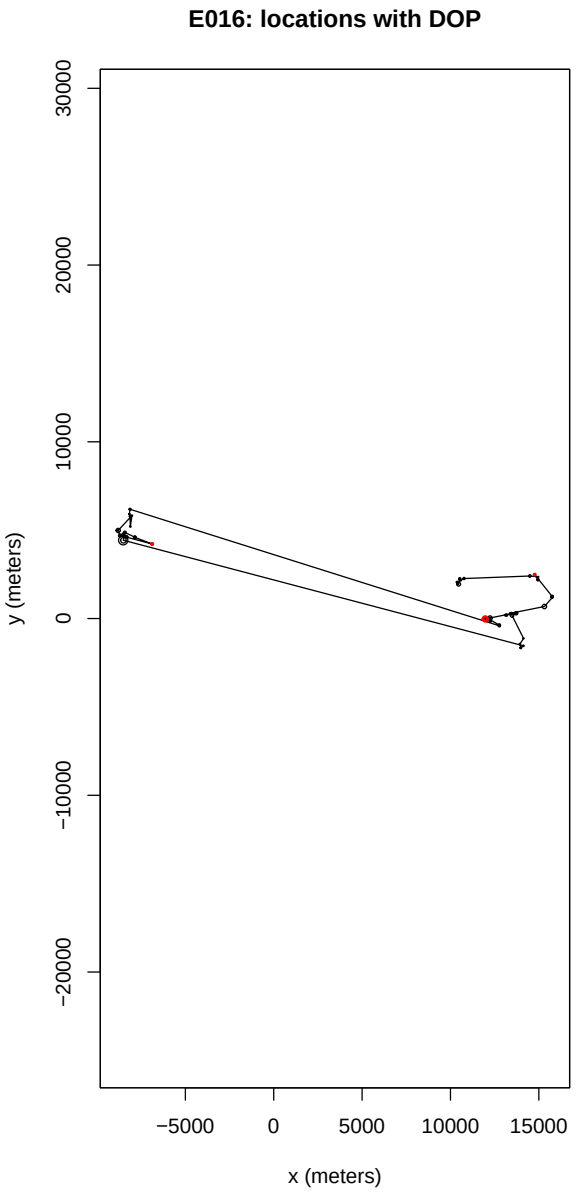


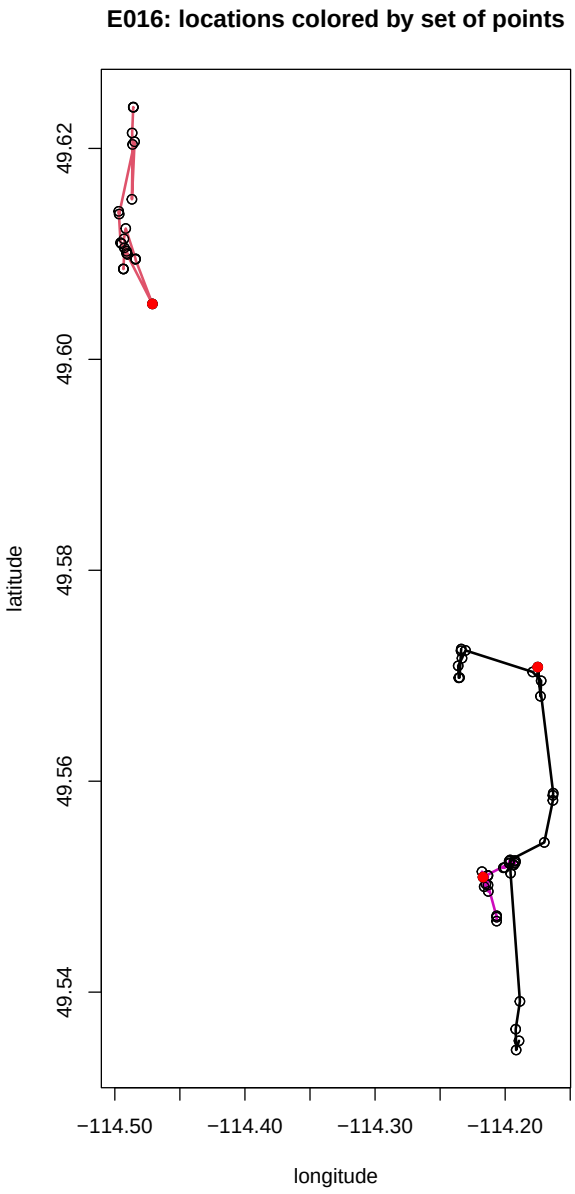




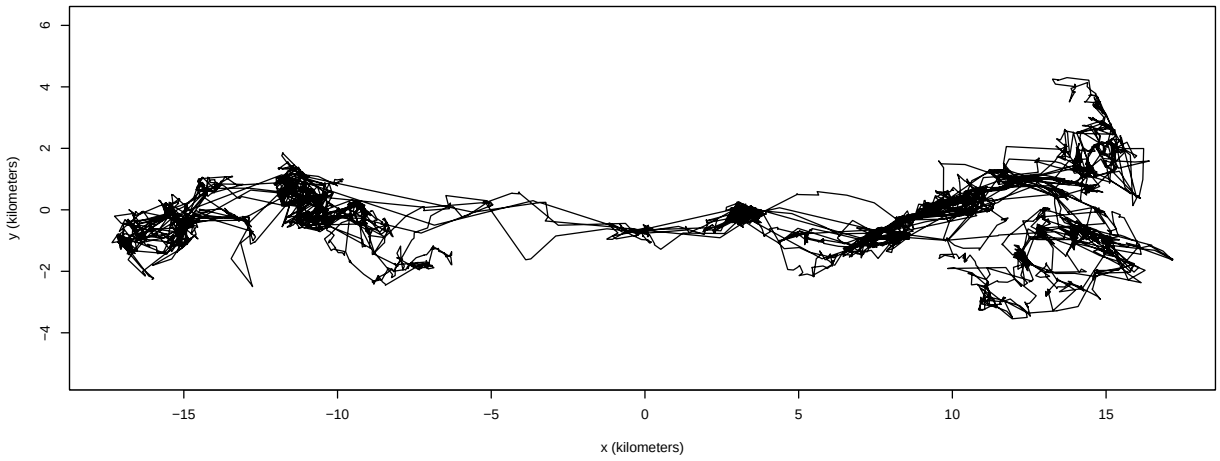


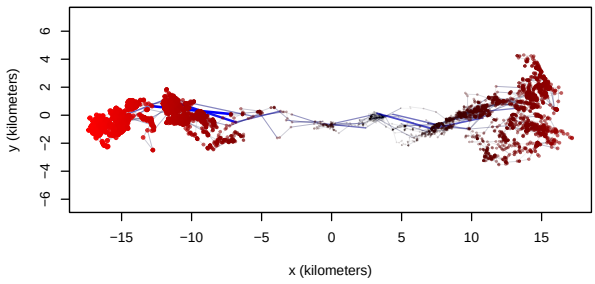

```
plot_adj('E016', max_speed = 0.2, max_angle = 170) # ok
```

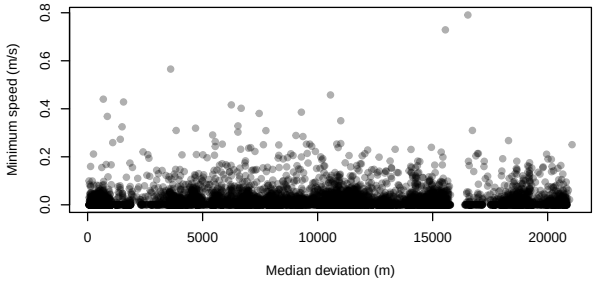


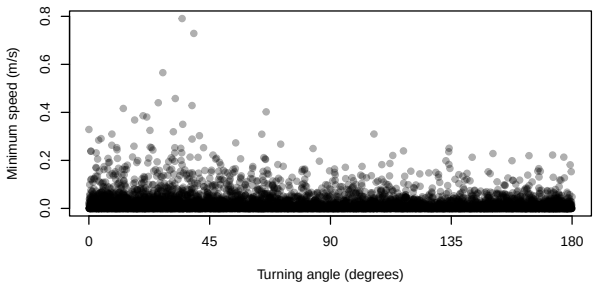


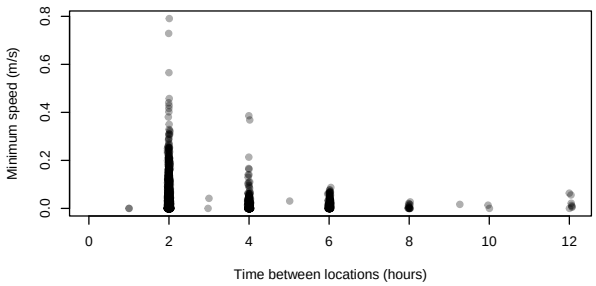

```
# E017
out <- check_animal('E017')
```



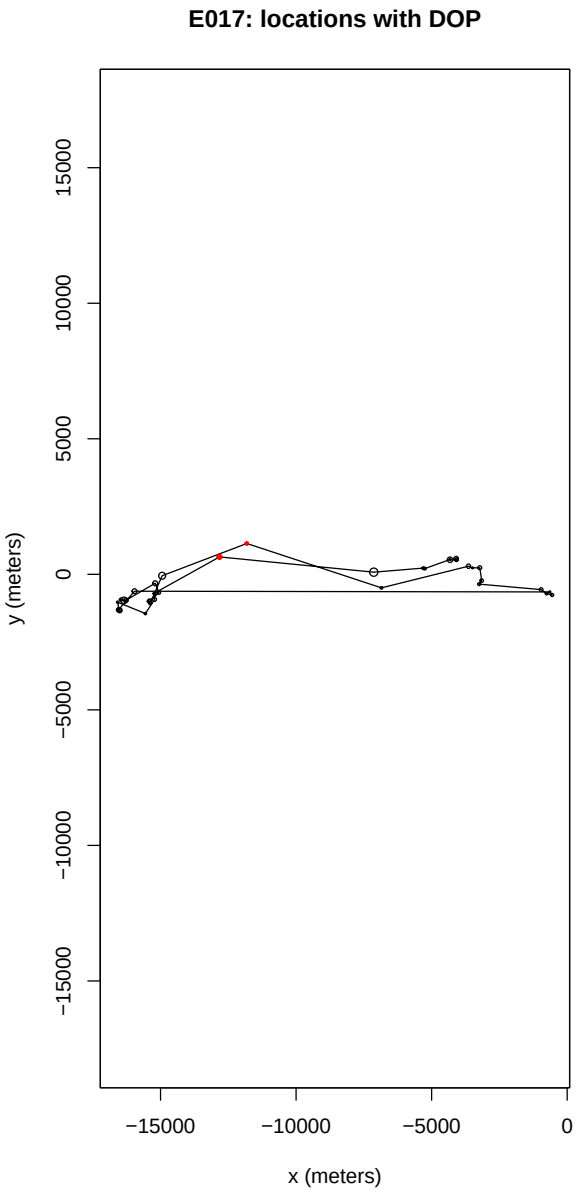


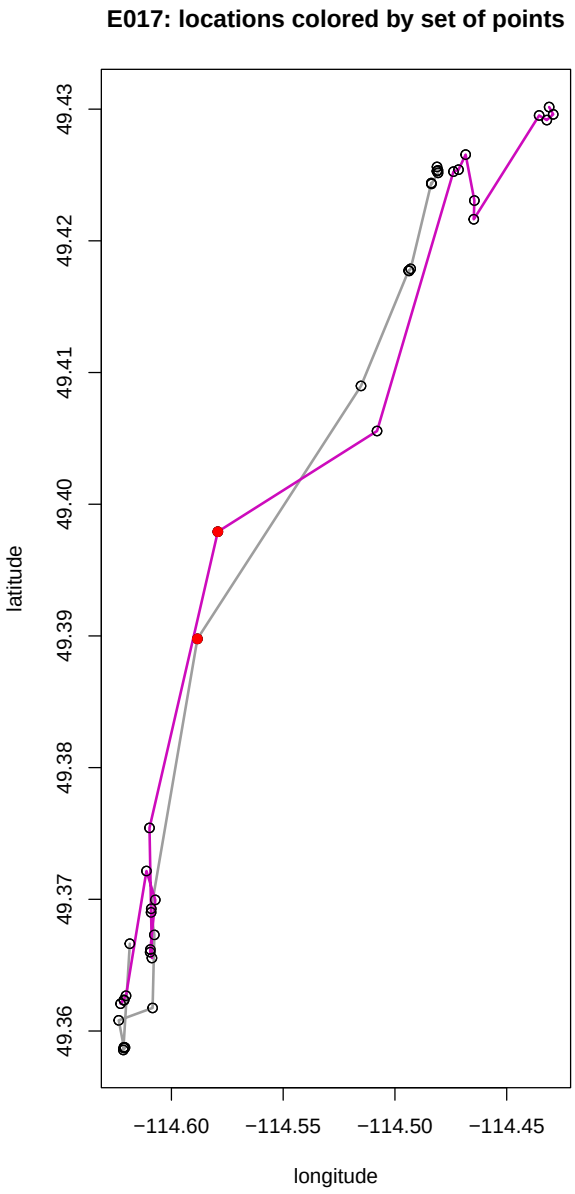




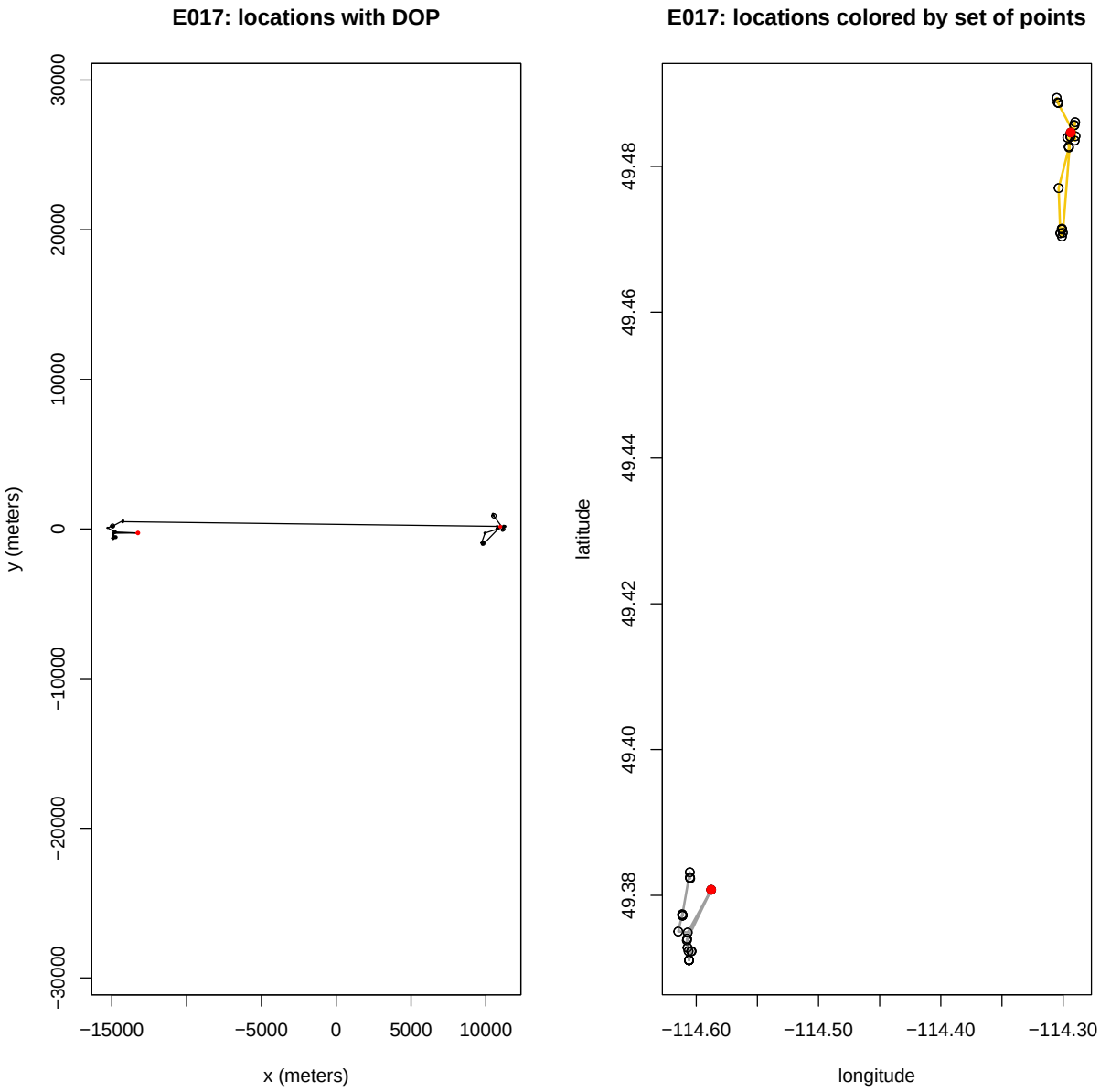



```
plot_adj('E017', max_speed = 0.6) # ok
```

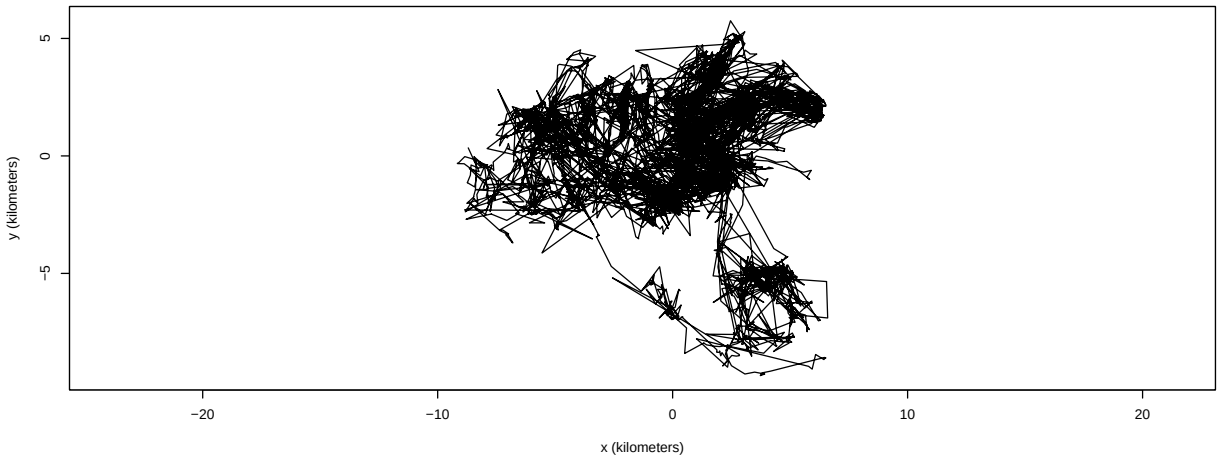


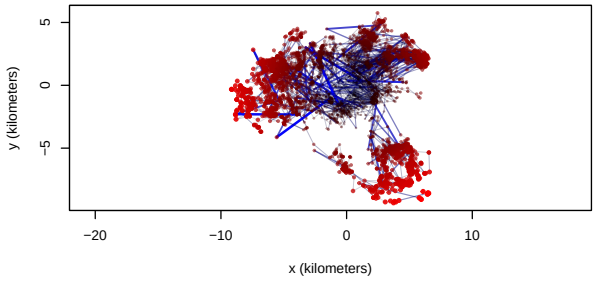


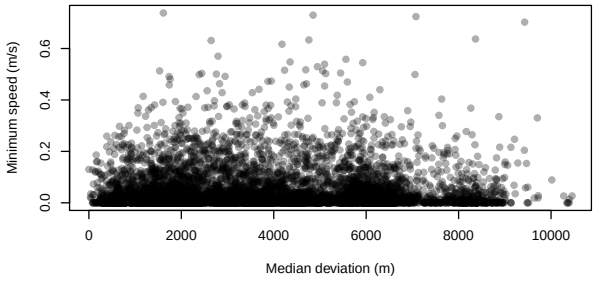

```
plot_adj('E017', max_angle = 170, max_speed = 0.2)
```

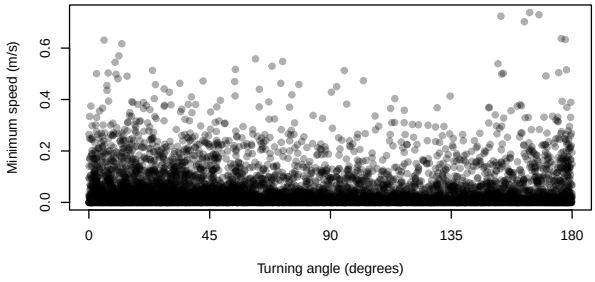


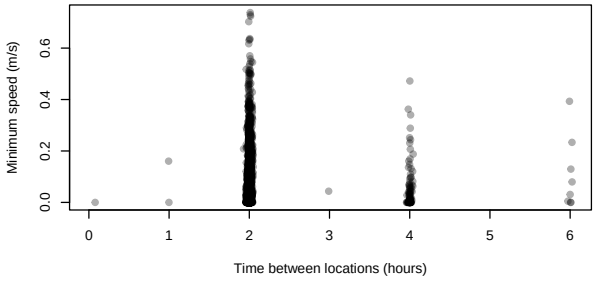

```
# E019
out <- check_animal('E019') # many high speeds at angle > 170
```



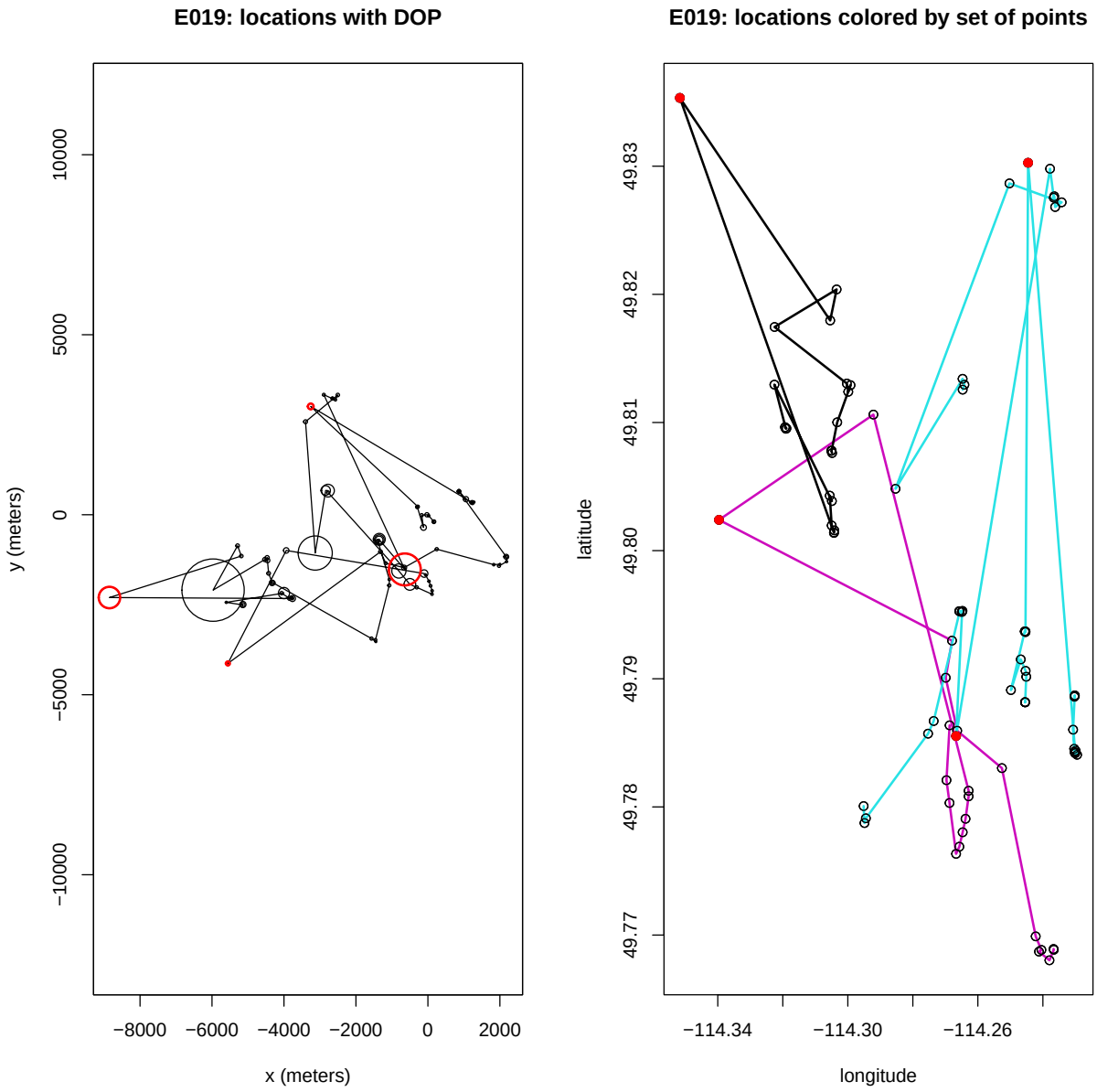




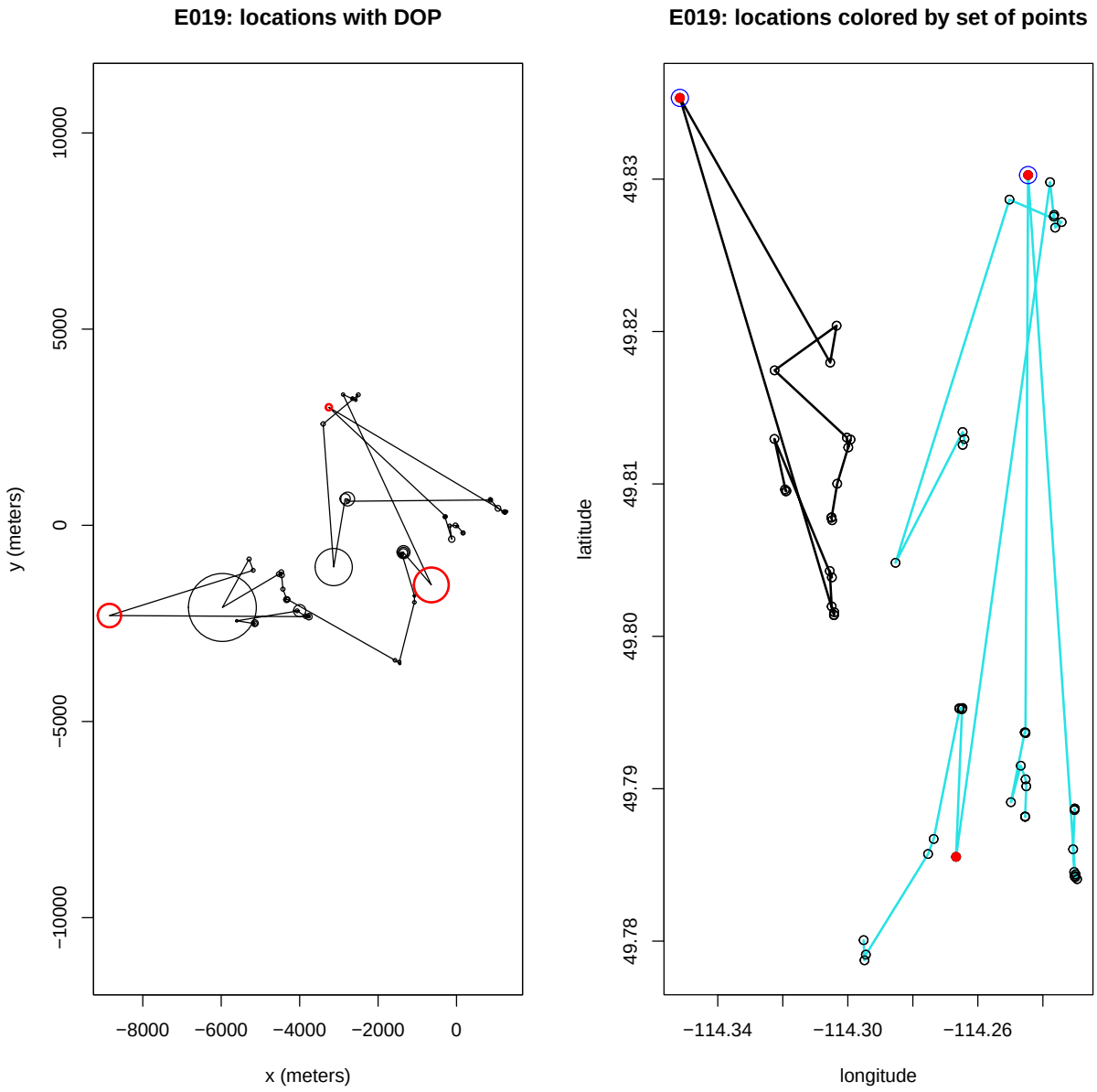




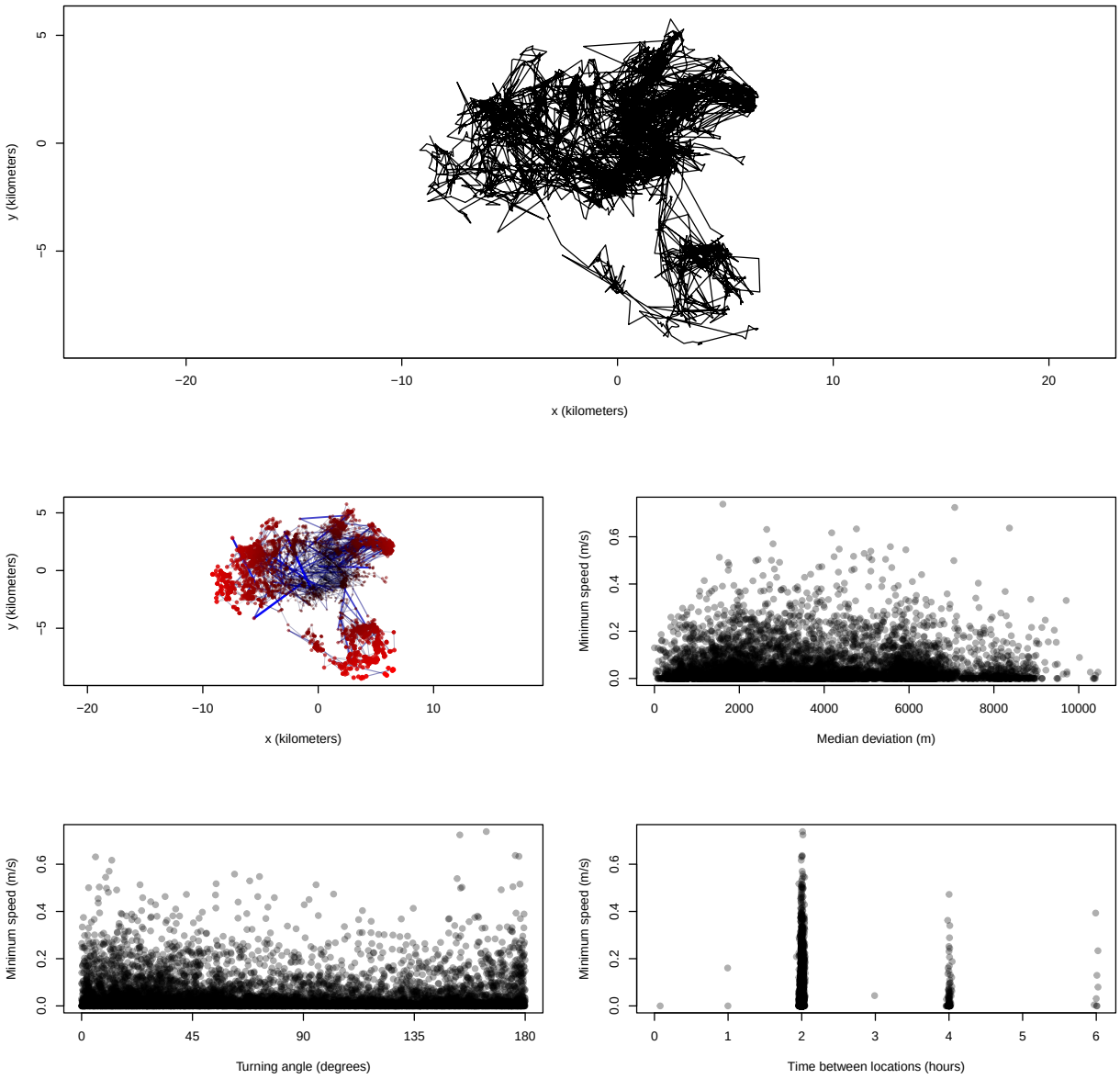

```
plot_adj('E019', max_speed = 0.65, max_angle = 135) # some outliers
```



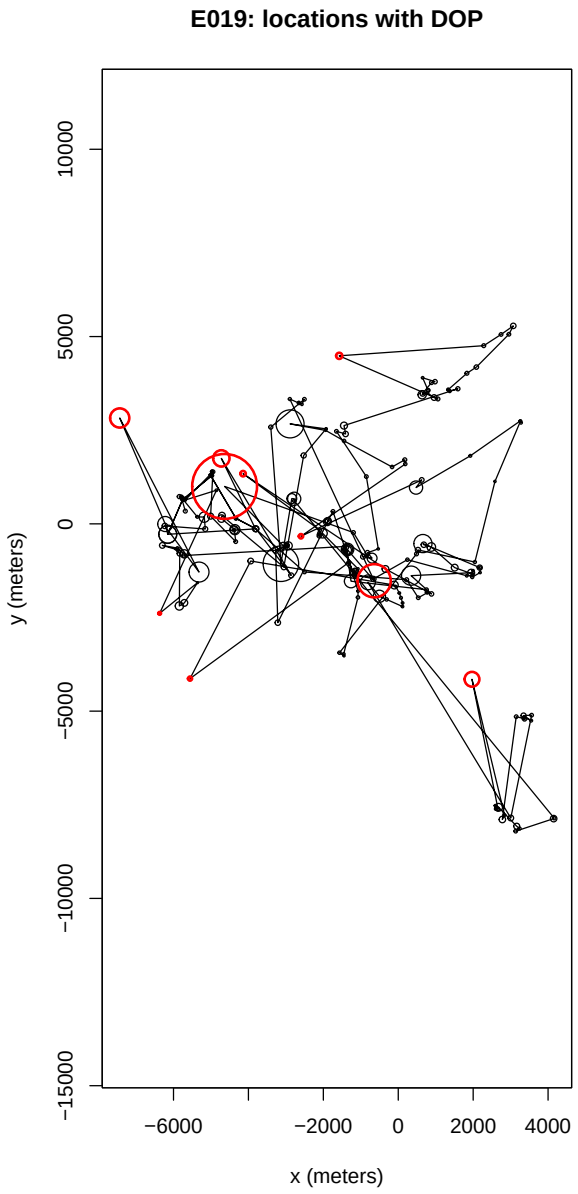

```
plot_adj('E019', max_speed = 0.65, max_angle = 160, reset_layout = FALSE)
i <- which(out$speed > 0.65 & out$angle > 160)[c(1, 3)] # some outliers
points(location.lat ~ location.long,
       d$tel[[which(d$animal == 'E019')]][i, ], cex = 2, col = 'blue')
```

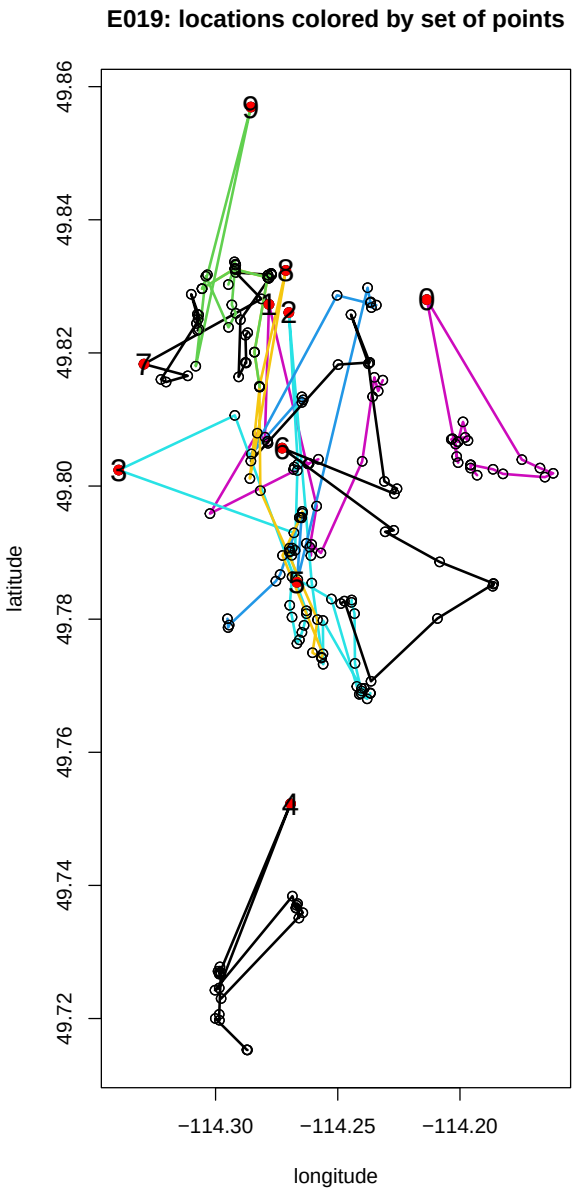



```
d$tel[[which(d$animal == 'E019')]][i, 'outlier'] <- 1

out <- check_animal('E019') # check other high speeds with angle > 160
```




```
plot_adj('E019', max_speed = 0.4, max_angle = 135, reset_layout = FALSE)
tel <- d$tel[[which(d$animal == 'E019')]]
i <- which(out$speed > 0.4 & out$angle > 135)
points(location.lat ~ location.long, cex = 1.5, col = 'black',
       tel[! tel$outlier, ][i, ],
       pch = as.character(0:(length(i) - 1)))
```



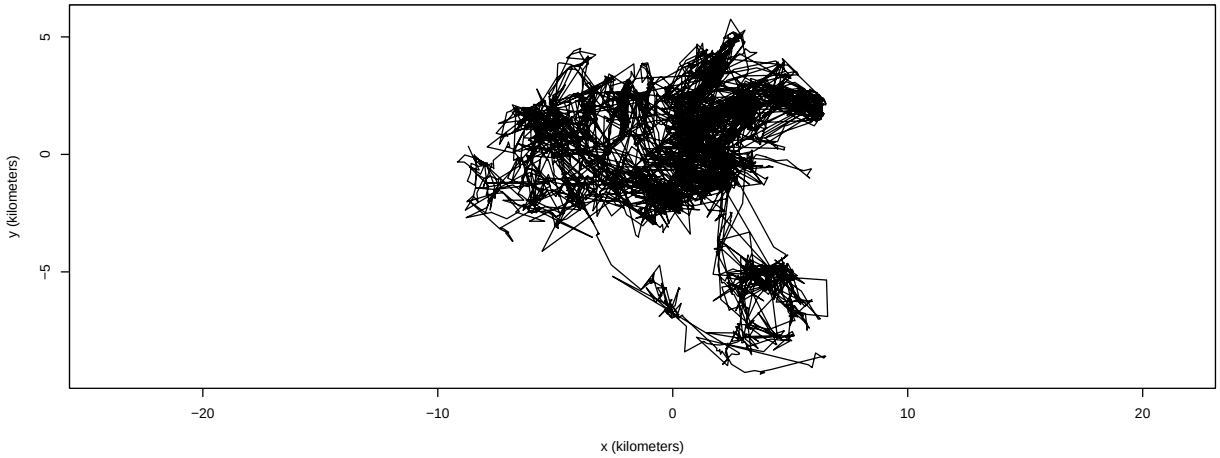


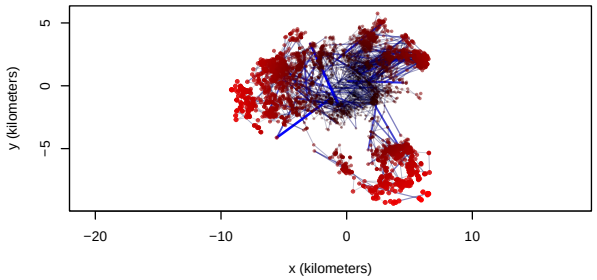

```
'hour' %#% out[i, 'dt'] # consistent sampling
```

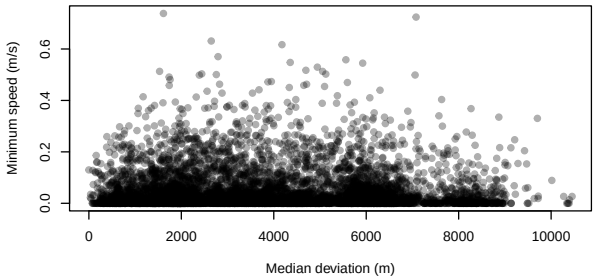
```
##  [1] 2.006667 2.010000 2.008333 2.016389 2.008333 2.011389 2.010278 2.015278
##  [9] 2.000278 2.005278
```

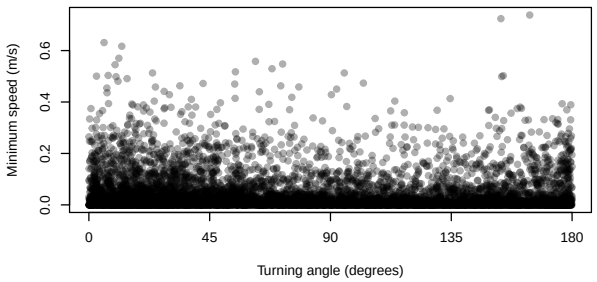
```
# 0 is likely an error (erring on the side of caution)
# 1 is consistent with other movement
# 2 is likely an error
# 3 is consistent with other movement
# 4 is likely an error
# 5 is realistic
# 6 is likely an error
# 7 is realistic
```

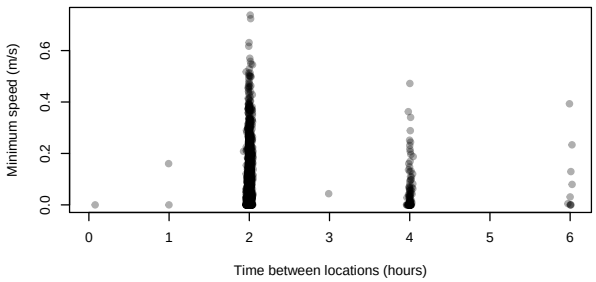
```r
# 8 is likely an error
# 9 is likely an error
i <- i[1 + c(0, 2, 4, 6, 8, 9)]
i_id <- which(d$animal == 'E019')
d$tel[[i_id]][! d$tel[[i_id]]$outlier, ][i, 'outlier'] <- 1
# ensure correct locations were removed
out <- check_animal('E019') # check other high speeds with angle > 160
```



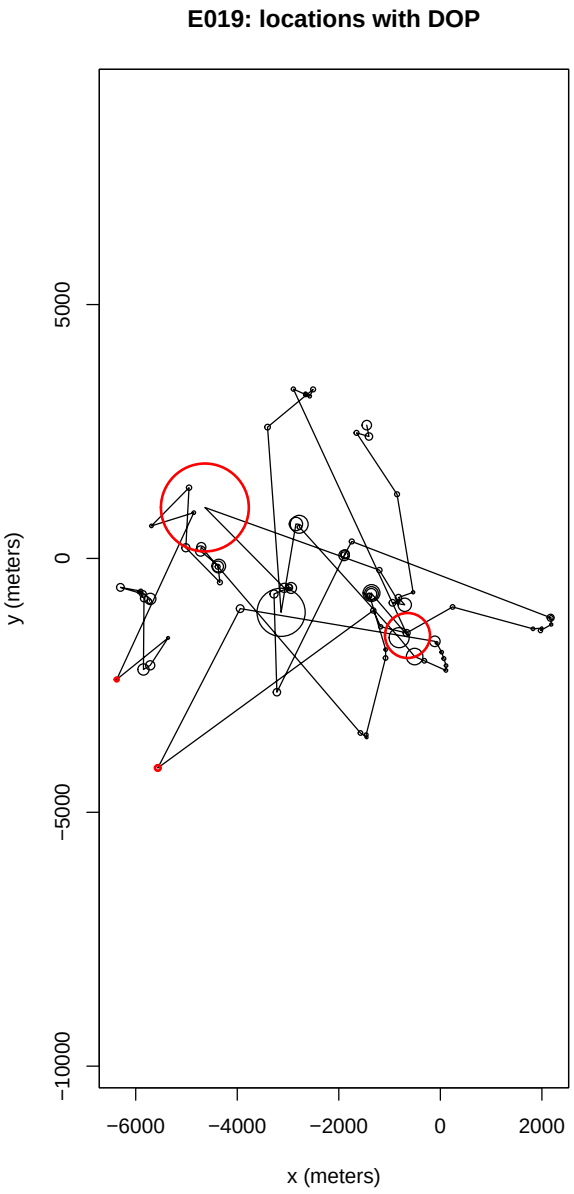


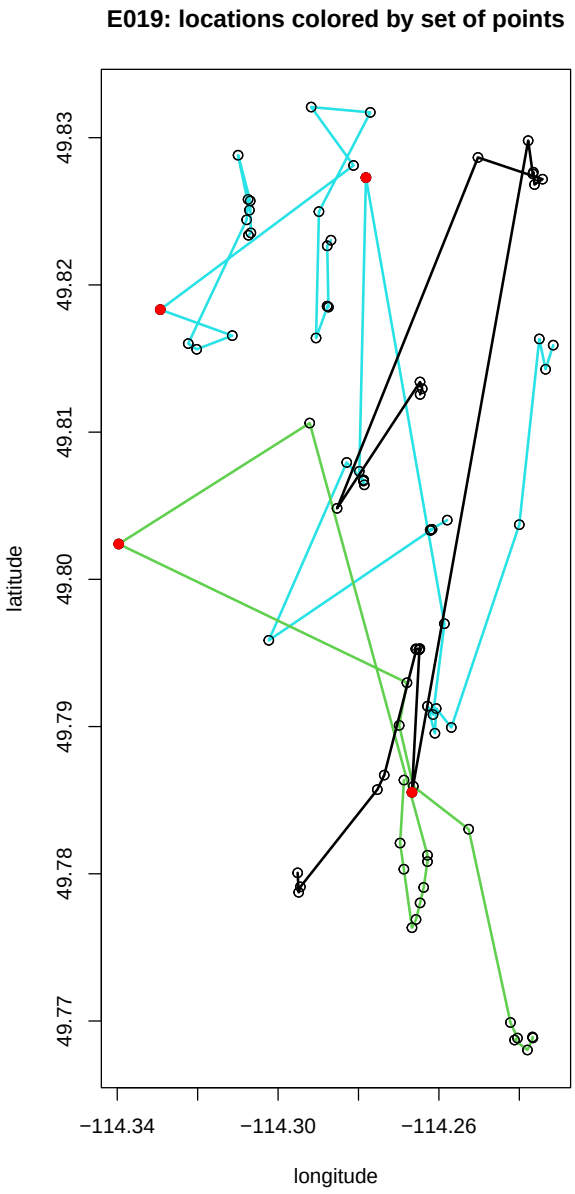







```r
plot_adj('E019', max_speed = 0.4, max_angle = 135, reset_layout = FALSE)
```


E019: locations with DOP
y (meters)
5000
0
−5000
−10000
−6000
−4000
−2000
0
2000
x (meters)


E019: locations colored by set of points
latitude
49.83
49.82
49.81
49.80
49.79
49.78
49.77
−114.34
−114.30
−114.26
longitude

References

Calabrese JM, Fleming CH, Gurarie E (2016) Ctmm: An R package for analyzing animal relocation data as a continuous-time stochastic process (ed Freckleton R). *Methods in Ecology and Evolution*, **7**, 1124–1132.

Nathan R, Monk CT, Arlinghaus R et al. (2022) Big-data approaches lead to an increased understanding of the ecology of animal movement. *Science*, **375**, eabg1780.

Noonan MJ, Fleming CH, Akre TS et al. (2019) Scale-insensitive estimation of speed and distance traveled from animal tracking data. *Movement Ecology*, **7**, 35.