# White-lipped peccary movement in PN Emas - exploratory and home range analysis

*Bernardo Niebuhr, Ennio Painkow, Ronaldo Morato - CENAP/ICMBio*

*2019-10-15*

## Loading and clean data

First of all, we load and organize data, to prepare them as input to the different movement analysis packages. We're working here with data from four white-lipped peccary individuals (from four different groups) from Parque Nacional das Emmas, within the Brazilian Cerrado.

```
# Set up

# Clean everything before beginning
rm(list = ls())

# Load packages
install.load::install_load('tidyverse', 'ggridges')
install.load::install_load('sf')
install.load::install_load('move', 'amt', 'adehabitatLT', 'ctmm')
```

### Load data

Below we load the data for each of the animals (separated in different sheets), transform dates and times into `datetime` objects, rename columns with easier names, and keep only the ones which are relevant for these analyses. In the end, we combine all `data.frame`s in a single one.

```
# Load Alto Forsmoso movement data
mov.data.alto.formoso <- readr::read_csv('data/6010 PLT Alto Formoso/Pontos Planilha 6010 PLT Alto Form
  dplyr::mutate(timestamp = lubridate::ymd_hms(`GPS Fix Time`),
                name = 'Alto Formoso') %>%
  dplyr::rename(fix_attempt = `GPS Fix Attempt`,
                latitude = `GPS Latitude`,
                longitude = `GPS Longitude`,
                utm.zone = `GPS UTM Zone`,
                utm.y = `GPS UTM Northing`,
                utm.x = `GPS UTM Easting`) %>%
  dplyr::select(name, timestamp, fix_attempt,
                longitude, latitude, utm.zone, utm.x, utm.y)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Acquisition Time` = col_character(),
##   `Acquisition Start Time` = col_character(),
##   `GPS Fix Time` = col_character(),
##   `GPS Fix Attempt` = col_character(),
##   `GPS UTM Zone` = col_character(),
##   `Satellite Uplink` = col_character(),
##   `Receive Time` = col_character(),
##   `Low Voltage` = col_character(),
```

```
##   Mortality = col_character(),
##   `Iridium Command` = col_logical(),
##   `Predeployment Data` = col_character(),
##   Error = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
mov.data.alto.formoso
```

```
## # A tibble: 4,780 x 8
##    name  timestamp           fix_attempt longitude latitude utm.zone  utm.x
##    <chr> <dttm>              <chr>           <dbl>    <dbl> <chr>     <dbl>
##  1 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  2 Alto~ 2019-04-03 00:28:02 Succeeded       -52.7    -18.3 22K      322842
##  3 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  4 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  5 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  6 Alto~ 2019-04-06 20:35:06 Succeeded       -52.8    -18.3 22K      312847
##  7 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  8 Alto~ NA                  <NA>             NA       NA   <NA>         NA
##  9 Alto~ NA                  <NA>             NA       NA   <NA>         NA
## 10 Alto~ 2019-04-07 14:16:37 Succeeded       -52.8    -18.3 22K      312834
## # ... with 4,770 more rows, and 1 more variable: utm.y <dbl>
```

```
# Load Olhos D'Agua Leste movement data
mov.data.olhos_dagua_leste <- readr::read_csv("data/Olhos Leste/data_log_olhos_leste.csv", skip = 22) %
  dplyr::mutate(timestamp = lubridate::ymd_hms(`GPS Fix Time`),
                name = "Olhos D'Agua Leste") %>%
  dplyr::rename(fix_attempt = `GPS Fix Attempt`,
                latitude = `GPS Latitude`,
                longitude = `GPS Longitude`,
                utm.zone = `GPS UTM Zone`,
                utm.y = `GPS UTM Northing`,
                utm.x = `GPS UTM Easting`) %>%
  dplyr::select(name, timestamp, fix_attempt,
                longitude, latitude, utm.zone, utm.x, utm.y)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Acquisition Time` = col_character(),
##   `Acquisition Start Time` = col_character(),
##   `GPS Fix Time` = col_character(),
##   `GPS Fix Attempt` = col_character(),
##   `GPS UTM Zone` = col_character(),
##   `GPS Satellite Bitmap` = col_character(),
##   `Satellite Uplink` = col_character(),
##   Mortality = col_character(),
##   `Iridium Command` = col_logical(),
##   `Predeployment Data` = col_character(),
##   Error = col_logical()
## )
## See spec(...) for full column specifications.
```

```
mov.data.olhos_dagua_leste
```

```
## # A tibble: 70,470 x 8
##    name  timestamp           fix_attempt longitude latitude utm.zone  utm.x
##    <chr> <dttm>              <chr>           <dbl>    <dbl> <chr>     <dbl>
##  1 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  2 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  3 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  4 Olho~ 2019-03-20 17:46:20 Succeeded ~     -112.     33.4 12S      424558
##  5 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  6 Olho~ 2019-03-20 18:00:25 Succeeded ~     -112.     33.4 12S      424555
##  7 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  8 Olho~ NA                  <NA>              NA       NA  <NA>         NA
##  9 Olho~ 2019-03-20 18:15:07 Succeeded ~     -112.     33.4 12S      424555
## 10 Olho~ NA                  <NA>              NA       NA  <NA>         NA
## # ... with 70,460 more rows, and 1 more variable: utm.y <dbl>
```

```
# Load Olhos D'Agua Oeste movement data
mov.data.olhos_dagua_oeste <- readr::read_csv("data/Olhos Oeste/Pontos Planilha Olhos Oeste Consolidado
  dplyr::mutate(timestamp = lubridate::ymd_hms(`GPS Fix Time`),
                name = "Olhos D'Agua Oeste") %>%
  dplyr::rename(fix_attempt = `GPS Fix Attempt`,
                latitude = `GPS Latitude`,
                longitude = `GPS Longitude`,
                utm.zone = `GPS UTM Zone`,
                utm.y = `GPS UTM Northing`,
                utm.x = `GPS UTM Easting`) %>%
  dplyr::select(name, timestamp, fix_attempt,
                longitude, latitude, utm.zone, utm.x, utm.y)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Acquisition Time` = col_character(),
##   `Acquisition Start Time` = col_character(),
##   `GPS Fix Time` = col_character(),
##   `GPS Fix Attempt` = col_character(),
##   `GPS UTM Zone` = col_character(),
##   `Satellite Uplink` = col_character(),
##   `Receive Time` = col_character(),
##   `Low Voltage` = col_character(),
##   Mortality = col_character(),
##   `Iridium Command` = col_logical(),
##   `Predeployment Data` = col_character(),
##   Error = col_character()
## )
## See spec(...) for full column specifications.
```

```
mov.data.olhos_dagua_oeste
```

```
## # A tibble: 6,119 x 8
##    name  timestamp           fix_attempt longitude latitude utm.zone  utm.x
##    <chr> <dttm>              <chr>           <dbl>    <dbl> <chr>     <dbl>
##  1 Olho~ NA                  <NA>             NA       NA  <NA>         NA
##  2 Olho~ 2019-03-31 21:12:57 Succeeded       -53.0    -18.3 22K      290981
```

```
##  3 Olho~ NA                  <NA>             NA       NA   <NA>         NA
##  4 Olho~ 2019-03-31 21:31:00 Succeeded       -53.0    -18.3 22K      290978
##  5 Olho~ 2019-03-31 22:00:19 Succeeded       -53.0    -18.3 22K      290980
##  6 Olho~ 2019-03-31 22:30:14 Succeeded       -53.0    -18.3 22K      290971
##  7 Olho~ 2019-03-31 23:00:28 Succeeded       -53.0    -18.3 22K      290957
##  8 Olho~ 2019-03-31 23:30:46 Succeeded       -53.0    -18.3 22K      290962
##  9 Olho~ 2019-04-01 00:01:55 Succeeded       -53.0    -18.3 22K      290958
## 10 Olho~ 2019-04-01 00:30:46 Succeeded       -53.0    -18.3 22K      290944
## # ... with 6,109 more rows, and 1 more variable: utm.y <dbl>
```

```
# Load Pontal movement data
mov.data.pontal <- readr::read_csv("data/Pontal/pontal_atualizado_20191011.csv", skip = 23) %>%
  dplyr::mutate(timestamp = lubridate::ymd_hms(`GPS Fix Time`),
                name = "Pontal") %>%
  dplyr::rename(fix_attempt = `GPS Fix Attempt`,
                latitude = `GPS Latitude`,
                longitude = `GPS Longitude`,
                utm.zone = `GPS UTM Zone`,
                utm.y = `GPS UTM Northing`,
                utm.x = `GPS UTM Easting`) %>%
  dplyr::select(name, timestamp, fix_attempt,
                longitude, latitude, utm.zone, utm.x, utm.y)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Acquisition Time` = col_character(),
##   `Acquisition Start Time` = col_character(),
##   `GPS Fix Time` = col_character(),
##   `GPS Fix Attempt` = col_character(),
##   `GPS UTM Zone` = col_character(),
##   `Satellite Uplink` = col_character(),
##   `Receive Time` = col_character(),
##   `Low Voltage` = col_character(),
##   Mortality = col_character(),
##   `Iridium Command` = col_logical(),
##   `Predeployment Data` = col_character(),
##   Error = col_logical()
## )
## See spec(...) for full column specifications.
```

```
## Warning: 12 parsing failures.
##  row   col           expected              actual                                          file
## 1441 Error 1/0/T/F/TRUE/FALSE Incomplete Transfer 'data/Pontal/pontal_atualizado_20191011.csv'
## 2860 Error 1/0/T/F/TRUE/FALSE Incomplete Transfer 'data/Pontal/pontal_atualizado_20191011.csv'
## 3303 Error 1/0/T/F/TRUE/FALSE Incomplete Transfer 'data/Pontal/pontal_atualizado_20191011.csv'
## 3927 Error 1/0/T/F/TRUE/FALSE Incomplete Transfer 'data/Pontal/pontal_atualizado_20191011.csv'
## 4913 Error 1/0/T/F/TRUE/FALSE Incomplete Transfer 'data/Pontal/pontal_atualizado_20191011.csv'
## .... ..... .................. ................... ..........................................
## See problems(...) for more details.
```

```
mov.data.pontal
```

```
## # A tibble: 9,282 x 8
##    name  timestamp           fix_attempt longitude latitude utm.zone  utm.x
##    <chr> <dttm>              <chr>           <dbl>    <dbl> <chr>     <dbl>
```

```
##  1 Pont~ NA                  <NA>             NA       NA   <NA>         NA
##  2 Pont~ 2019-04-03 22:02:01 Succeeded       -53.0    -18.4 22K      290837
##  3 Pont~ NA                  <NA>             NA       NA   <NA>         NA
##  4 Pont~ NA                  <NA>             NA       NA   <NA>         NA
##  5 Pont~ 2019-04-03 22:30:37 Succeeded       -53.0    -18.4 22K      290838
##  6 Pont~ 2019-04-03 23:00:07 Succeeded       -53.0    -18.4 22K      290837
##  7 Pont~ 2019-04-03 23:30:48 Succeeded       -53.0    -18.4 22K      290838
##  8 Pont~ 2019-04-04 00:00:47 Succeeded       -53.0    -18.4 22K      290827
##  9 Pont~ 2019-04-04 00:30:48 Succeeded       -53.0    -18.4 22K      290818
## 10 Pont~ 2019-04-04 01:00:46 Succeeded       -53.0    -18.4 22K      290825
## # ... with 9,272 more rows, and 1 more variable: utm.y <dbl>
```

```
# Merge data
mov.data <- dplyr::bind_rows(mov.data.alto.formoso, mov.data.olhos_dagua_leste, mov.data.olhos_dagua_oes
```

## Explore and filter data

Now we perform a general data cleaning. We count the number of failed attempts to get data and remove them, remove duplicate points, remove spikes (extreme ouliers, in a more or less manual way), plot the data (just to check if everything seems right), export them as a single sheet, but also as Shapefile and Geopackage files, to be open directly in GIS software. We also standardize the starting date of the data - we consider only locations after April 8th 2019.

```
# Total number of fails in fix attempt
table(mov.data$fix_attempt)
```

```
##
##         Failed      Succeeded Succeeded (2D) Succeeded (3D)
##           5090          16787             17           6700
```

```
sum(mov.data$fix_attempt == 'Failed', na.rm = TRUE)
```

```
## [1] 5090
```

```
# Proportion of fails
100*sum(mov.data$fix_attempt == 'Failed', na.rm = TRUE)/nrow(mov.data)
```

```
## [1] 5.614941
```

```
# Per individual
mov.data %>%
  group_by(name) %>%
  summarize(
    number_of_fails = sum(fix_attempt == 'Failed', na.rm = TRUE),
    proportion_of_fails = 100*sum(fix_attempt == 'Failed', na.rm = TRUE)/n()
  ) %>%
  kable()
```

| name | number_of_fails | proportion_of_fails |
|---|---|---|
| Alto Formoso | 148 | 3.096234 |
| Olhos D'Agua Leste | 2028 | 2.877820 |
| Olhos D'Agua Oeste | 2827 | 46.200359 |
| Pontal | 87 | 0.937298 |

```
# Remove data with missing or failed fixed attempt, arrange per name and date
mov.data.clean <- mov.data %>%
```

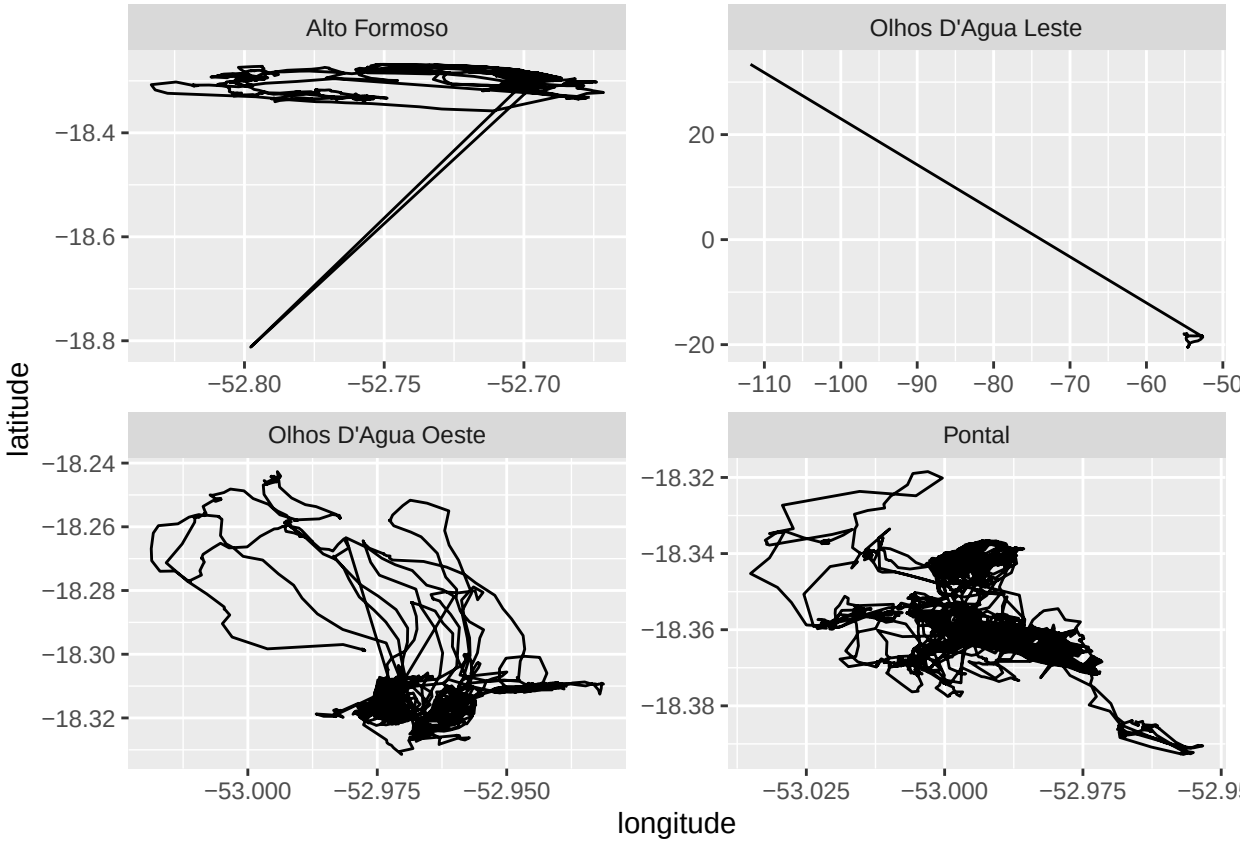
```
  dplyr::filter(!is.na(fix_attempt), fix_attempt != 'Failed') %>%
  dplyr::arrange(name, timestamp)

# Remove duplicates
dupl <- mov.data.clean %>%
  dplyr::select(name, timestamp) %>%
  duplicated
sum(dupl)
```

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

```
mov.data.clean[dupl,]
```

```
## # A tibble: 0 x 8
## # ... with 8 variables: name <chr>, timestamp <dttm>, fix_attempt <chr>,
## #   longitude <dbl>, latitude <dbl>, utm.zone <chr>, utm.x <dbl>,
## #   utm.y <dbl>
```

```
mov.data.clean <- mov.data.clean[!dupl,]

# Plot
ggplot(data = mov.data.clean) +
  geom_path(aes(x = longitude, y = latitude)) +
  facet_wrap(~ name, scales = 'free')
```

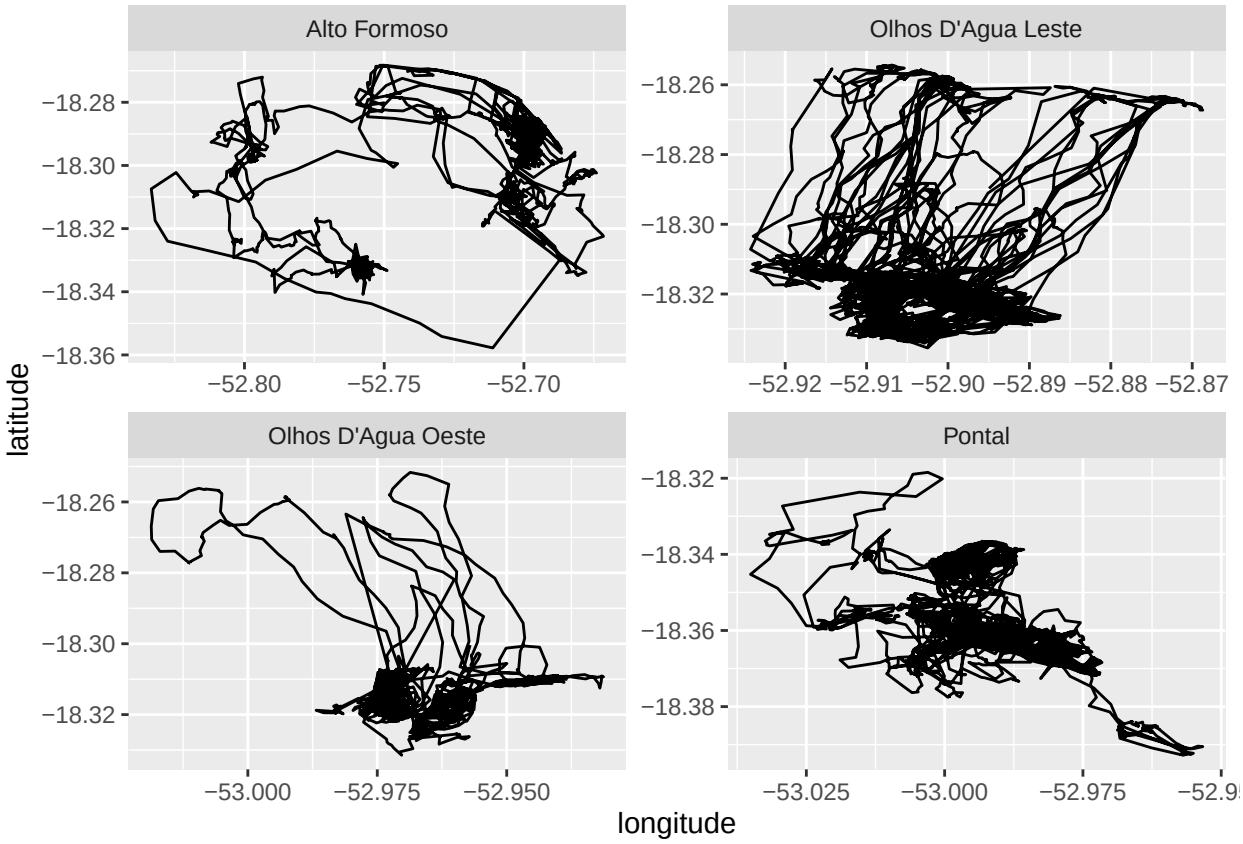



```
# There seem to be outliers, let's remove them
# First we romeve the first points
```

```
mov.data.clean <- mov.data.clean %>%
  dplyr::filter(timestamp > lubridate::ymd_hm('2019-04-08 00:00'))

# Remove outliers
(names <- unique(mov.data.clean$name))
```

```
## [1] "Alto Formoso"       "Olhos D'Agua Leste" "Olhos D'Agua Oeste"
## [4] "Pontal"
```

```
out.af <- which(mov.data.clean$name == names[1] & mov.data.clean$latitude < -18.6)
out.odal <- which(mov.data.clean$name == names[2] & (mov.data.clean$longitude < -52.94 | mov.data.clean$

mov.data.clean <- mov.data.clean[-c(out.af, out.odal),]

# Re-Plot
ggplot(data = mov.data.clean) +
  geom_path(aes(x = longitude, y = latitude)) +
  facet_wrap(~ name, scales = 'free')
```




```
# Export as vector
mov.data.clean %>%
  sf::st_as_sf(coords = c('longitude', 'latitude'), crs = 4326) %>% # CRS(+proj=longlat +ellps=WGS84 +d
  sf::st_write('data/movement_data_all_individuals.shp', delete_dsn = TRUE)
```

```
## Warning in abbreviate_shapefile_names(obj): Field names abbreviated for
## ESRI Shapefile driver
```

```
## Deleting source `data/movement_data_all_individuals.shp' using driver `ESRI Shapefile'
## Writing layer `movement_data_all_individuals' to data source `data/movement_data_all_individuals.shp
```

```
## Warning in CPL_write_ogr(obj, dsn, layer, driver,
## as.character(dataset_options), : GDAL Message 6: Field timstmp create as
## date field, though DateTime requested.
```

```
## Writing 22649 features with 6 fields and geometry type Point.
```

```
mov.data.clean %>%
  sf::st_as_sf(coords = c('longitude', 'latitude'), crs = 4326) %>% # CRS(+proj=longlat +ellps=WGS84 +d
  sf::st_write(paste0('data/movement_data_all_individuals.gpkg'),
               delete_dsn = TRUE)
```

```
## Deleting source `data/movement_data_all_individuals.gpkg' using driver `GPKG'
## Updating layer `movement_data_all_individuals' to data source `data/movement_data_all_individuals.gpk
## Writing 22649 features with 6 fields and geometry type Point.
```

```
mov.data.clean %>%
  readr::write_csv('data/movement_data_all_individuals.csv')
```

## General information on monitoring time and movement parameters

In this section we transform data into a `track` object from the `amt` package and plot information about the monitoring time (and exclude some sparse data from one of the individuals). We also calculate and plot some basic movement parameters, such as step lengths and turning angles, for individual and for different time frames.

```
# amt
mov.track <- mov.data.clean %>%
  amt::mk_track(.x = longitude, .y = latitude, .t = timestamp, crs = sp::CRS("+init=epsg:4326"),
                name)
```

```
## .t found, creating `track_xyt`.
```

```
# movement basic statistics
mov.track %>%
  group_by(name) %>%
  summarise(
    begin = min(t_),
    end = max(t_),
    range = diff(range(t_)),
    n = n()
  ) %>%
  kable
```

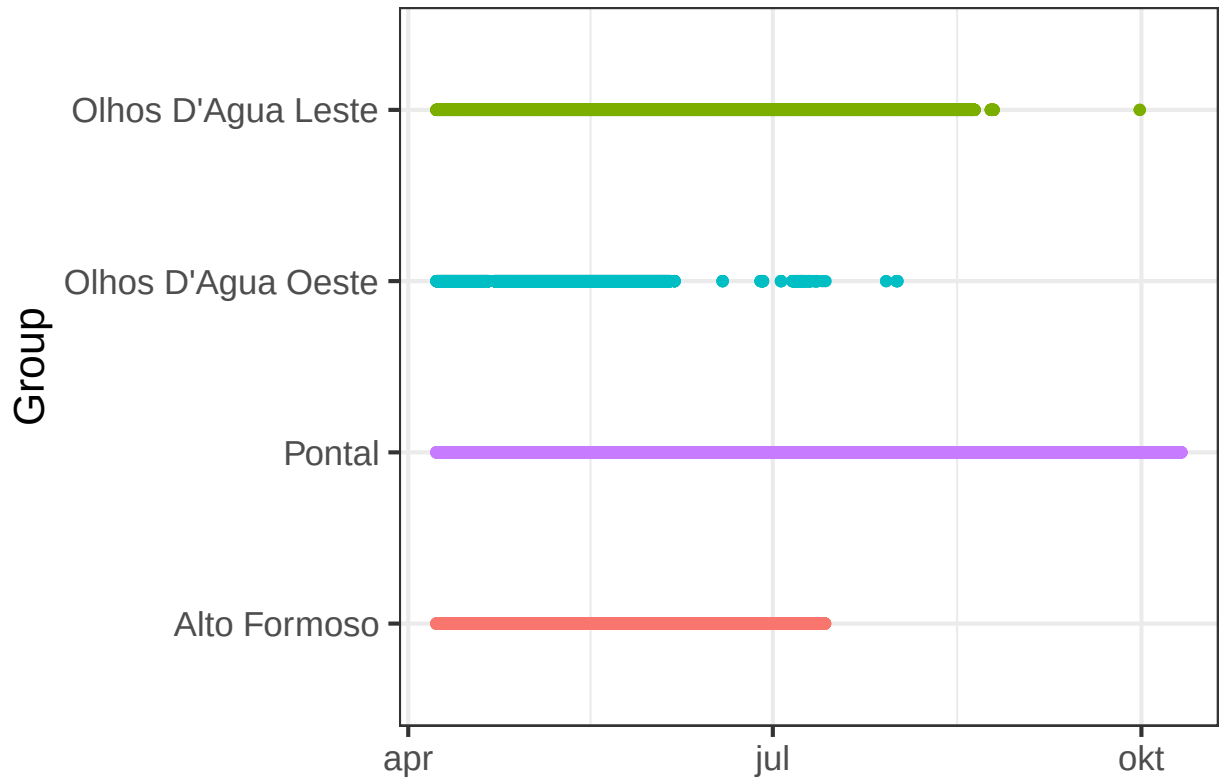
| name | begin | end | range | n |
|---|---|---|---|---|
| Alto Formoso | 2019-04-08 00:00:47 | 2019-07-14 01:30:47 | 97.0625 days | 4525 |
| Olhos D'Agua Leste | 2019-04-08 00:00:09 | 2019-09-30 14:30:11 | 175.6042 days | 6410 |
| Olhos D'Agua Oeste | 2019-04-08 00:00:09 | 2019-08-01 02:30:37 | 115.1045 days | 2872 |
| Pontal | 2019-04-08 00:00:33 | 2019-10-11 00:00:35 | 186.0000 days | 8842 |

```
# Monitoring period
mov.track
```

```
## # A tibble: 22,649 x 4
```

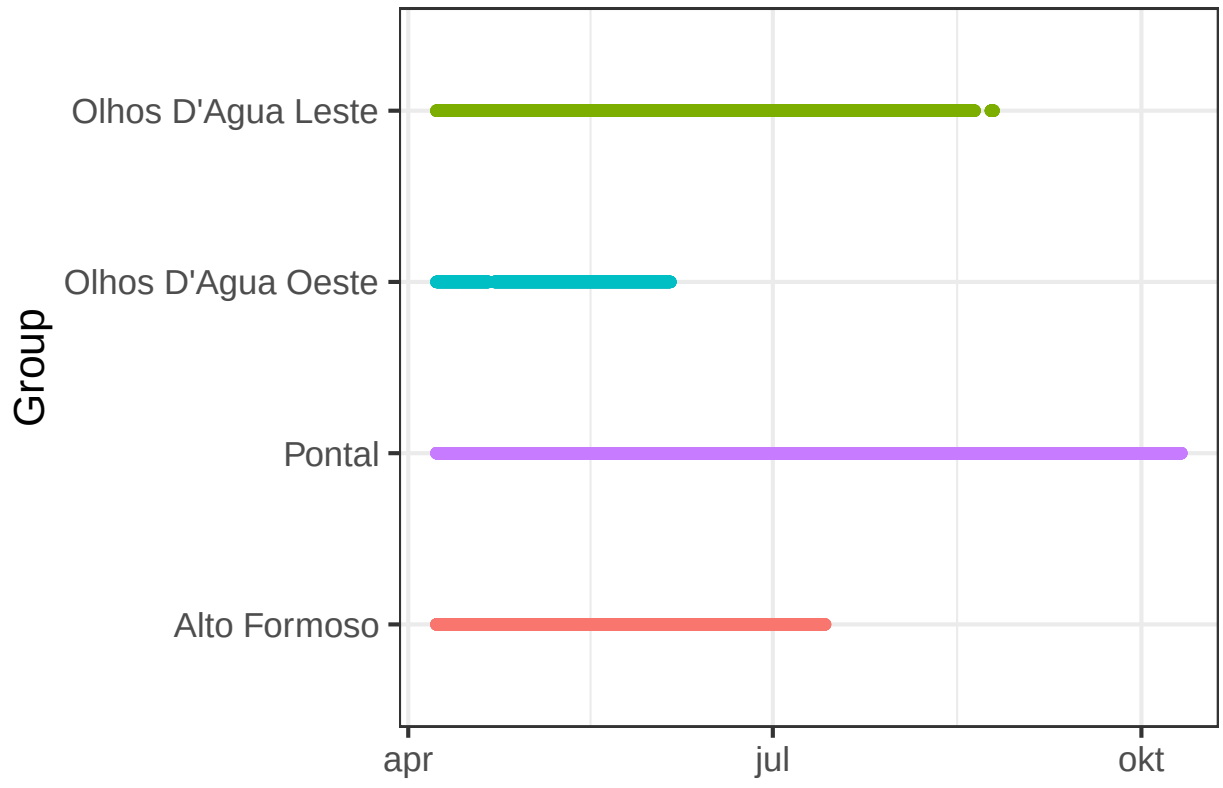
```
##       x_    y_ t_                  name
##   * <dbl> <dbl> <dttm>              <chr>
##   1 -52.9 -18.3 2019-04-08 00:00:09 Olhos D'Agua Leste
##   2 -53.0 -18.3 2019-04-08 00:00:09 Olhos D'Agua Oeste
##   3 -53.0 -18.4 2019-04-08 00:00:33 Pontal
##   4 -52.8 -18.3 2019-04-08 00:00:47 Alto Formoso
##   5 -52.9 -18.3 2019-04-08 00:30:09 Olhos D'Agua Leste
##   6 -53.0 -18.3 2019-04-08 00:30:09 Olhos D'Agua Oeste
##   7 -53.0 -18.4 2019-04-08 00:30:28 Pontal
##   8 -52.8 -18.3 2019-04-08 00:30:46 Alto Formoso
##   9 -53.0 -18.3 2019-04-08 01:00:11 Olhos D'Agua Oeste
##  10 -53.0 -18.4 2019-04-08 01:00:15 Pontal
## # ... with 22,639 more rows
```

```
g.mon <- ggplot() +
  #geom_vline(xintercept = as.POSIXct('2019-01-01'), linetype = 2) +
  geom_point(data = mov.track, aes(x = t_, y = factor(name, levels = rev(unique(mov.track$name))), col =
  theme_bw(base_size = 16) +
  theme(legend.position = "none") +
  #scale_x_datetime(labels = date_format("%m/%y"), limits = as.POSIXct(c('2018-06-01', '2019-04-01')),
  #                 date_minor_breaks = '1 month', date_breaks = '2 months') +
  labs(x = '',
       y = 'Group')
g.mon
```




**Should we consider the movement of 'Olhos D'Agua Oeste' only until the beginning of June?**
For now, let's do that. We also remove data from Olhos D'Agua Leste after the beginning of September.

```
# first we do that for the original cleaned data
mov.data.clean <- mov.data.clean %>%
  dplyr::filter(!(name == names[3] & timestamp > lubridate::ymd('2019-06-06')) & !(name == names[2] & t

# now we do the same for the track object
mov.track <- mov.track %>%
  dplyr::filter(!(name == names[3] & t_ > lubridate::ymd('2019-06-06')) & !(name == names[2] & t_ > lubr

# replot
# Monitoring period
g.mon <- ggplot() +
  #geom_vline(xintercept = as.POSIXct('2019-01-01'), linetype = 2) +
  geom_point(data = mov.track, aes(x = t_, y = factor(name, levels = rev(unique(mov.track$name))), col =
  theme_bw(base_size = 16) +
  theme(legend.position = "none") +
  #scale_x_datetime(labels = date_format("%m/%y"), limits = as.POSIXct(c('2018-06-01', '2019-04-01')),
  #                 date_minor_breaks = '1 month', date_breaks = '2 months') +
  labs(x = '',
       y = 'Group')
g.mon
```




```
# Calculaet basic movement parameters
mov.track.st <- mov.track %>%
  amt::transform_coords('+init=epsg:32722') %>% # transform to utm 21S
  group_by(name) %>%
  nest %>%
```

```
  mutate(sl_ = map(data, function(x) amt::step_lengths(x)/1000),
         ta_ = map(data, amt::direction_rel),
         aa_ = map(data, amt::direction_abs)) %>%
  unnest()
```

```
## Warning: `cols` is now required.
## Please use `cols = c(data, sl_, ta_, aa_)`
```

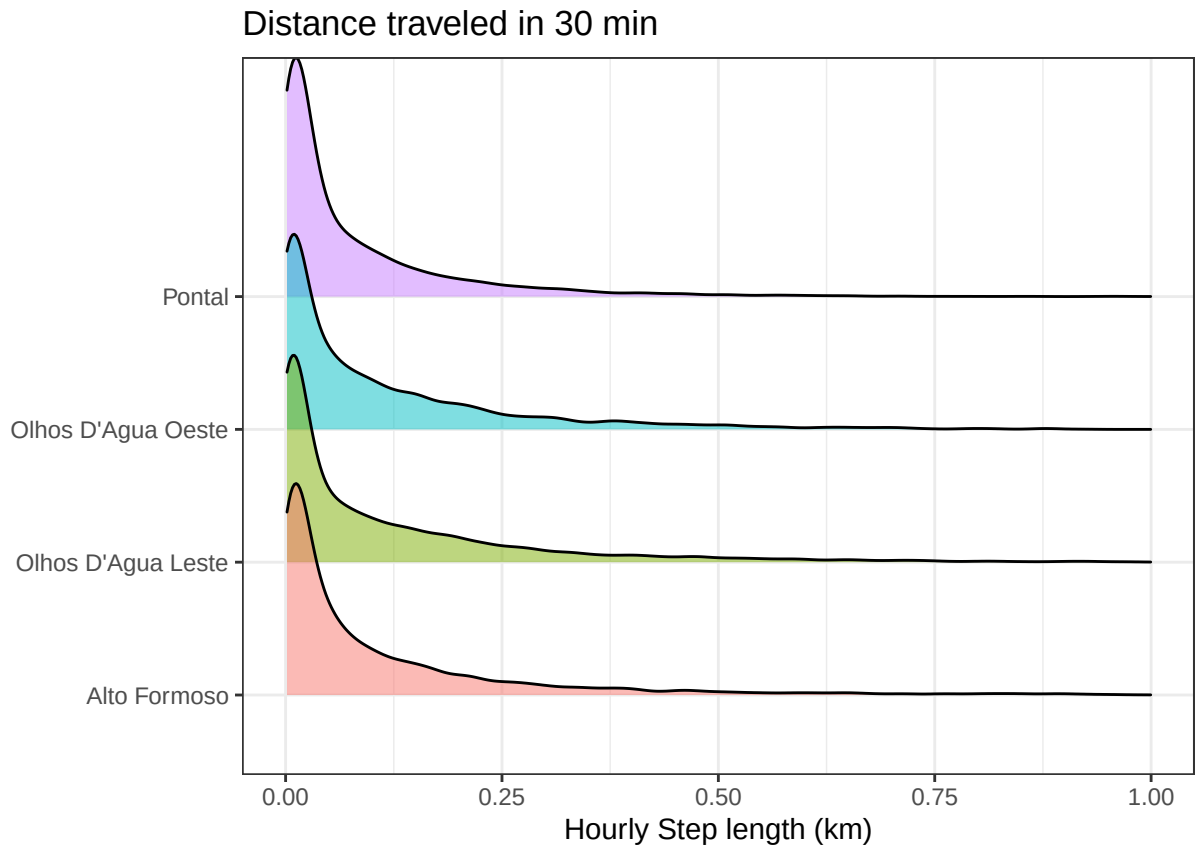
```
# amt::time_of_day()
```

### Average distance per hour and per day

```
# Step length histogram
g1 <- ggplot(data = mov.track.st, aes(x = sl_, y = name, fill = name)) +
  ggridges::geom_density_ridges(alpha = 0.5) +
  theme_bw() +
  labs(x = 'Hourly Step length (km)', y = '',
       title = 'Distance traveled in 30 min') +
  theme(legend.position = "none") +
  xlim(0, 1)
g1
```

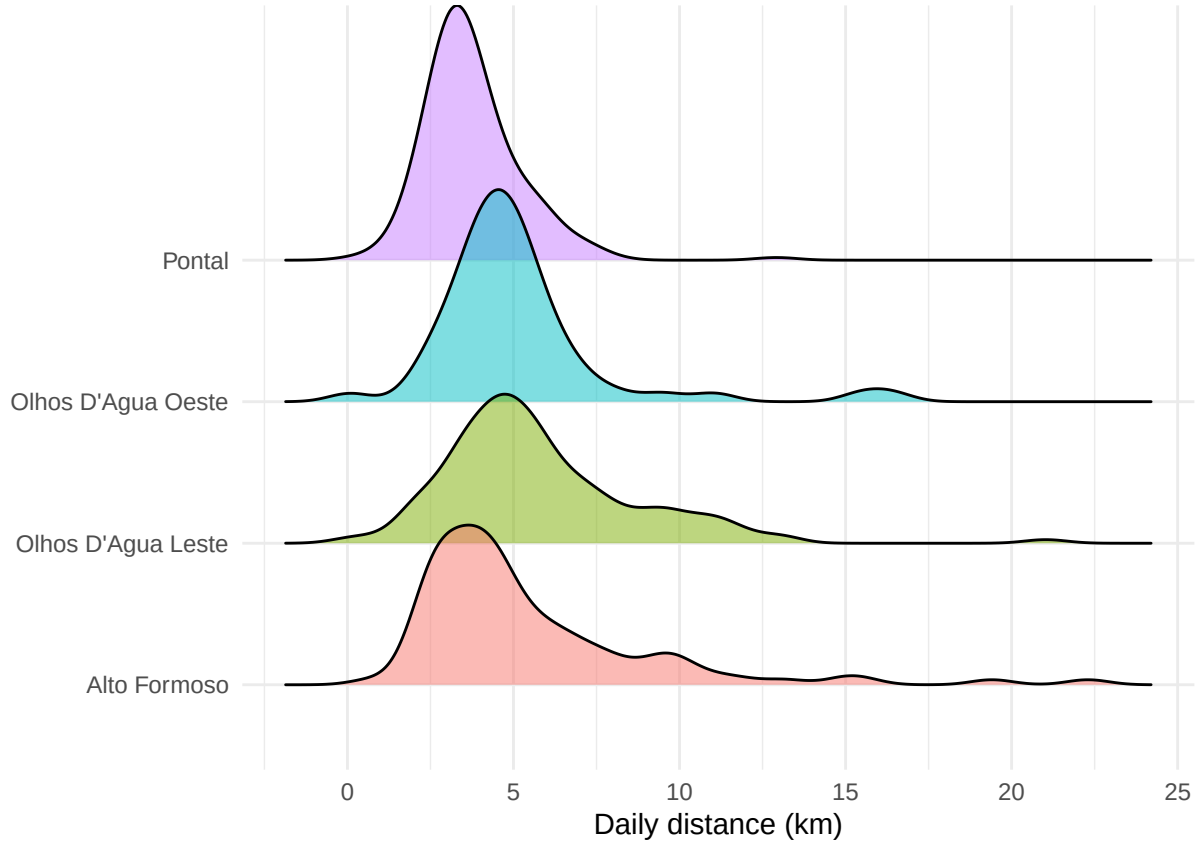
```
## Picking joint bandwidth of 0.0159
```

```
## Warning: Removed 111 rows containing non-finite values
## (stat_density_ridges).
```



```r
# Daily distance
mov.track.st$julian <- round(julian(mov.track.st$t_), 0)

mov.per.day <- mov.track.st %>%
  group_by(name, julian) %>%
  summarise(
    daily_distance = sum(sl_, na.rm = T),
    mean_angle = mean(ta_, na.rm = T)
  ) %>%
  mutate(mean_speed = daily_distance/24)

g2 <- ggplot(data = mov.per.day, aes(x = daily_distance, y = name, fill = name)) +
  geom_density_ridges(alpha = 0.5) +
  theme_minimal() +
  labs(x = 'Daily distance (km)', y = '') +
  theme(legend.position = "none")
g2
```

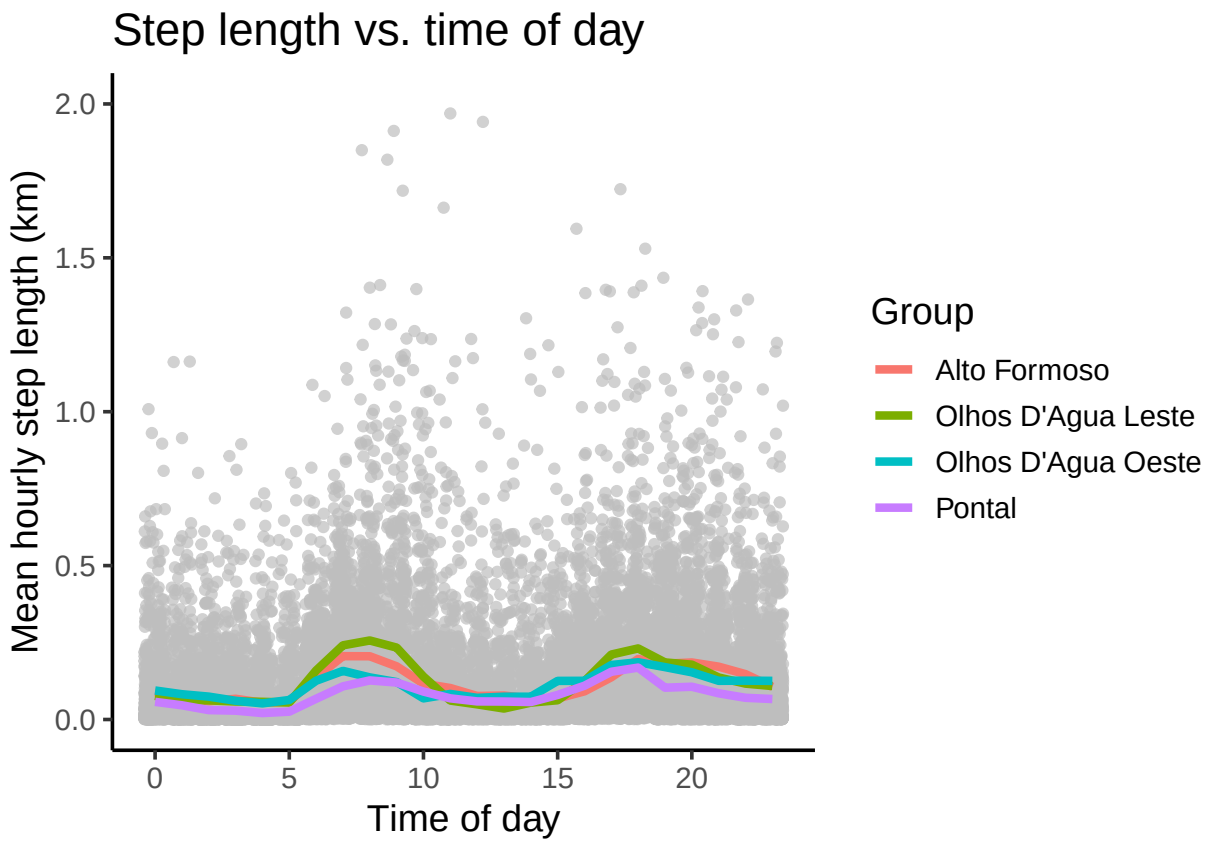
```
## Picking joint bandwidth of 0.635
```




```r
# Distance vs. time of day (removing 3hr from time)
mov.per.tod <- mov.track.st %>%
  mutate(hour = t_ - hours(3),
         hour = lubridate::hour(hour))
mov.per.tod.avg <- mov.per.tod %>%
  group_by(name, hour) %>%
```

```
  summarise(
    hourly_distance = mean(sl_, na.rm = T),
    mean_angle = mean(ta_, na.rm = T)
  )

g.tod <- ggplot() +
  geom_jitter(data = mov.per.tod, aes(x = hour, y = sl_), alpha = 0.7, col = 'grey') +
  geom_line(data = mov.per.tod.avg, aes(x = hour, y = hourly_distance, col = name, group = name),
            size = 1.5) +
  theme_classic(base_size = 14) +
  labs(x = 'Time of day',
       y = 'Mean hourly step length (km)',
       title = 'Step length vs. time of day',
       col = 'Group') +
  ylim(0, 2)
g.tod
```

```
## Warning: Removed 62 rows containing missing values (geom_point).
```



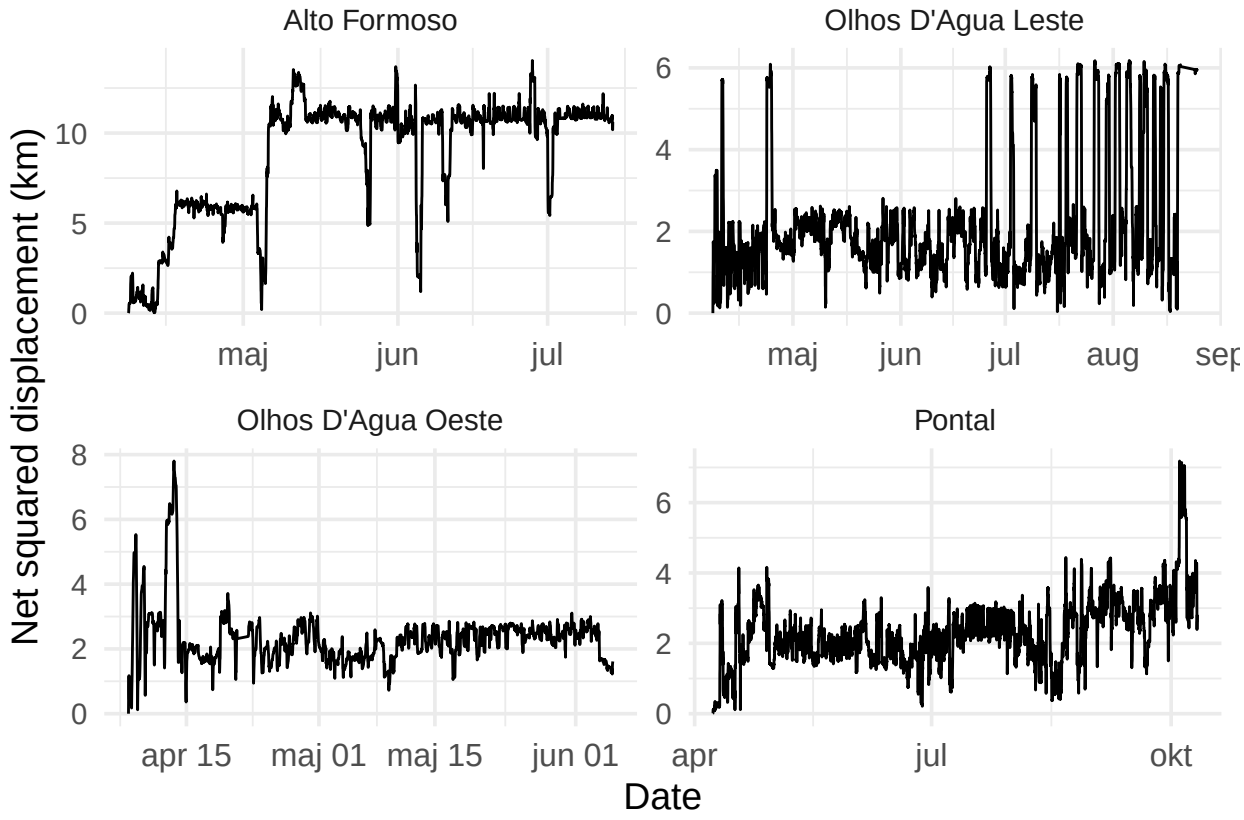

## Are animals in a home range behavior?

Next step is to understand whether animals present a home range behavior, if they are still in some kind of transient state to a ranging stability, or show some trace that resembmles dispersal or nomadic movement. To do so, first we calculate and plot the net square displacement along time, for each individual, to keep track of how they move from their initial position, and if they seem to keep in the same areas.

### Method 1) Net square displacement

From the two analytical approaches below, it seems animals definetely present a ranging behavior. Maybe the animal from Alto Formoso still presents a bias towards long distance travels, but we consider here this animal as ranging also. Therefore we runn `ctmm` analyses for all of them, to calculate their areas of use based on their variograms, continuous time movement models, and the autocorrelated kernel density estimator (AKDE).

```
# Calculate nsd
mov.track.nsd <- mov.track.st %>%
  tidyr::nest(data = -name) %>%
  dplyr::mutate(nsd = map(data, function(x) sqrt((x$x_ - x$x_[1])**2 + (x$y_ - x$y_[1])**2))) %>%
  tidyr::unnest(cols = c(data, nsd))

g3 <- ggplot(mov.track.nsd) +
  geom_line(aes(x = t_, y = nsd/1000)) +
  #geom_line(data = mov.traj.df.Min, aes(x = date, y = sqrt(R2n)/1000), col = 2) +
  facet_wrap(~ name, scales = 'free') +
  theme_minimal(base_size = 14) +
  labs(x = 'Date', y = 'Net squared displacement (km)') +
  theme(axis.text.x = element_text(size = 12))
g3
```




Here we save all data loaded, just to be able to use it already organized in other analyses.

```
# save workspace
save.image('data/movement_data_loaded.RData')
```

### Method 2) `ctmm` variograms

To run ctmm analyses, first we transform originalk cleaned data into a move object, from the package `move` and reproject it to the UTM 22S projection. Then we transform it into a `telemetry` object so that it can be used for analyses in the `ctmm` package.

Next step is to plot the variograms for each individual, as a second way of checking whether the animals present a ranging behavior. Then we fit the continuous time movement models to the variograms, and use them to calculate the AKDE. Finally we plot them and also export the 99% AKDE isopleths, as a basis for defining the area available to animals to understand their resource selection behavior.

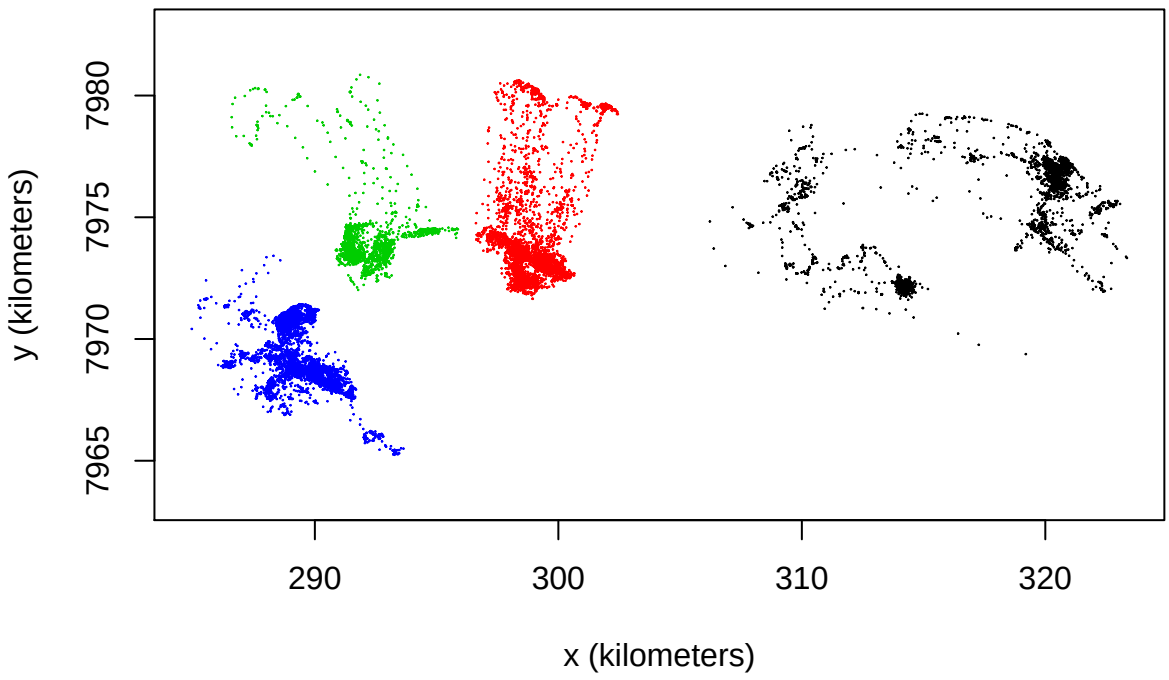
```
# Transform data into move object
mov.data.clean.df <- as.data.frame(mov.data.clean)
move.data <- move::move(x = mov.data.clean.df$longitude, y = mov.data.clean.df$latitude,
                        time = mov.data.clean.df$timestamp,
                        proj = sp::CRS("+init=epsg:4326"),
                        animal = mov.data.clean.df$name, sensor = 'GPS',
                        data = mov.data.clean.df)
move.data
```

```
## class       : MoveStack
## features    : 22504
## extent      : -53.03509, -52.67152, -18.39277, -18.25165  (xmin, xmax, ymin, ymax)
## crs         : +init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## variables   : 6
## names       :  timestamp,    fix_attempt,  longitude,   latitude,  utm.x,   utm.y
## min values  : 1554681609,      Succeeded, -53.035088, -18.392771, 284947, 7965245
## max values  : 1570752035, Succeeded (3D),  -52.67152, -18.251655, 323353, 7980852
## timestamps  : 2019-04-08 00:00:09 ... 2019-10-11 00:00:35 Time difference of 186 days  (start ... en
## sensors     : GPS
## indiv. data : name, utm.zone
## min ID Data : Alto Formoso, 22K
## max ID Data :       Pontal, 22K
## individuals : Alto.Formoso, Olhos.D.Agua.Leste, Olhos.D.Agua.Oeste, Pontal
## date created: 2019-10-08 00:11:46
```

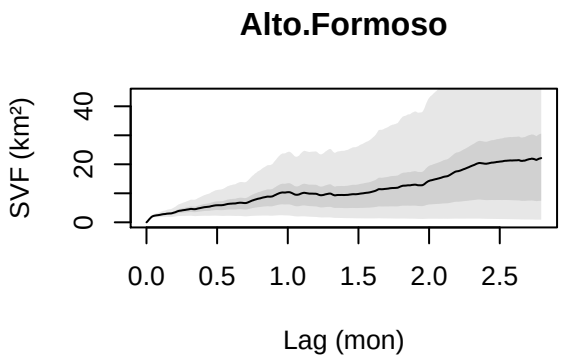
```
move.data.meters <- spTransform(move.data, CRSobj = '+init=epsg:32722')

# Prepare data in the ctmm format
mov.tel <- as.telemetry(move.data.meters, projection = move.data.meters@proj4string)
```

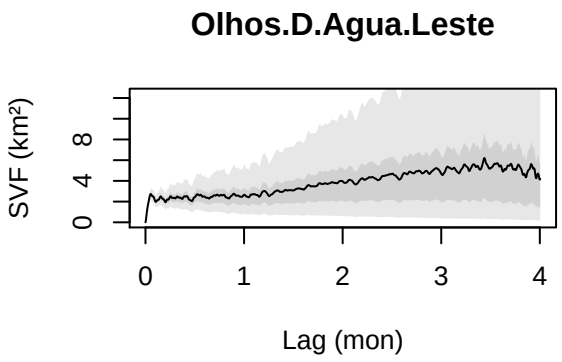
```
## Minimum sampling interval of 27.2 minutes in Alto Formoso

## Minimum sampling interval of 27.2 minutes in Olhos D'Agua Leste

## Minimum sampling interval of 27.6 minutes in Olhos D'Agua Oeste

## Minimum sampling interval of 27.4 minutes in Pontal
```

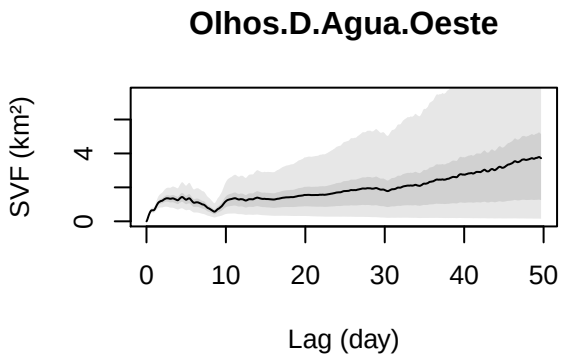
```
# Plot
plot(mov.tel, col = 1:length(mov.tel), pch = 19)
```

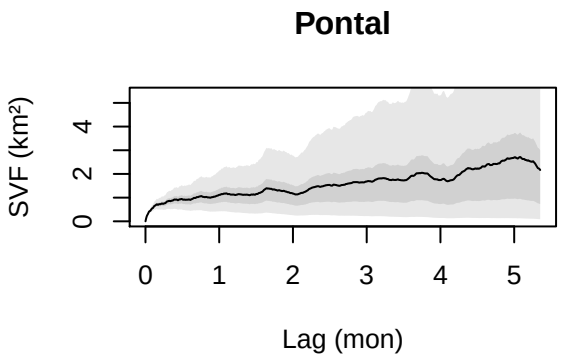



```
par(mfrow = c(2,2))
SVF <- list()
for(i in 1:length(mov.tel)) {
  if(length(mov.tel) == 1) {
    ind <- mov.tel
  } else {
    ind <- mov.tel[[i]]
  }
  SVF[[i]] <- variogram(ind)
  level <- c(0.5,0.95) # 50% and 95% CIs

  main.title <- names(mov.tel[i])
  xlim <- c(0, abs(difftime(ind$timestamp[1], ind$timestamp[length(ind$timestamp)], units = 'days')/30))
  plot(SVF[[i]], fraction = 0.85, level=level, axes = T)
  title(main.title)
}
```



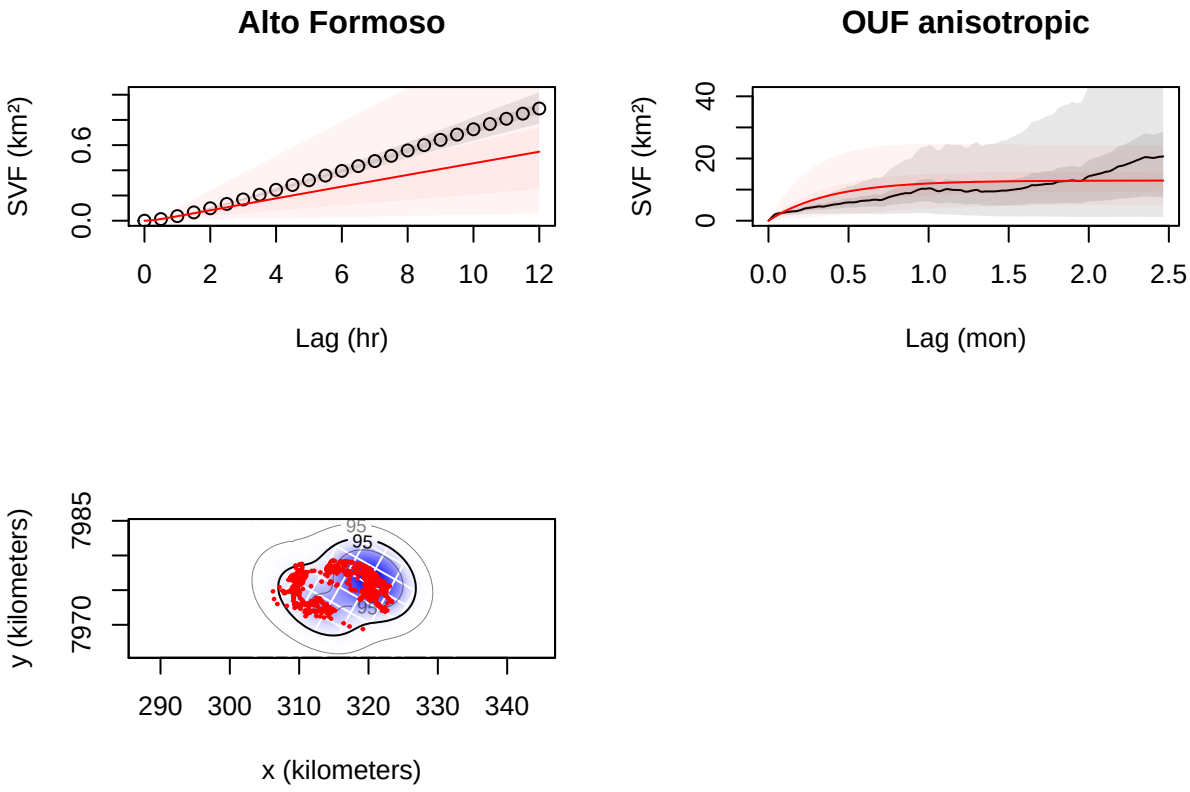




```
# akde analysis
FITS.all <- list()
AKDE.all <- list()

for(i in 1:length(mov.tel)) {
  print(paste('individual', i, sep = ' = '))

  ind <- mov.tel[[i]]
  SVF <- variogram(ind)

  level <- c(0.5, 0.95) # 50% and 95% CIs
  xlim <- c(0, 12 %#% "hour") # 0-12 hour window

  m.ouf <- ctmm.guess(ind, interactive=FALSE) # automated model guess
  FITS <- ctmm.select(ind, m.ouf, verbose=TRUE, level=1)
  FITS.all[[i]] <- FITS

  par(mfrow = c(2,2))
  plot(SVF, CTMM = FITS[[1]], xlim=xlim, level=level)
  title(ind@info$identity)
  plot(SVF, CTMM = FITS[[1]], fraction = 0.75, level=level)
  title(names(FITS[1]))

  HR <- akde(ind, CTMM = FITS[[1]])
  plot(ind, UD = HR)
  #title(rownames(summary(FITS))[1])
```

```
  AKDE.all[[i]] <- HR

  # krig <- occurrence(ind, CTMM = FITS[[1]])
  # plot(krig)
}
```

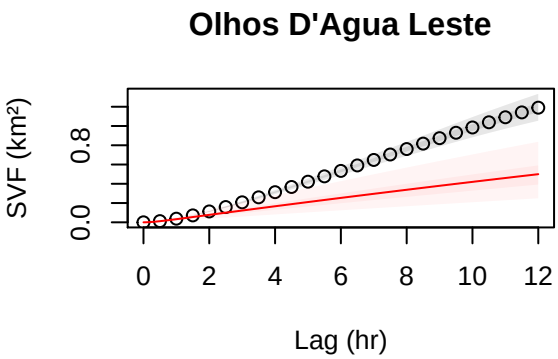
```
## [1] "individual = 1"
```

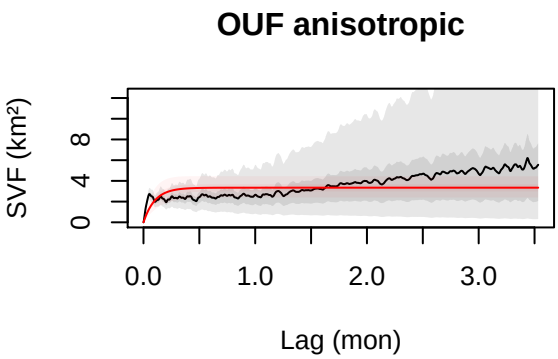


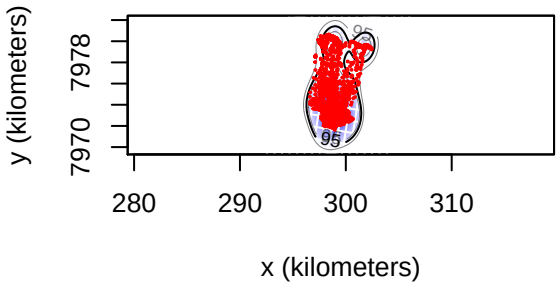




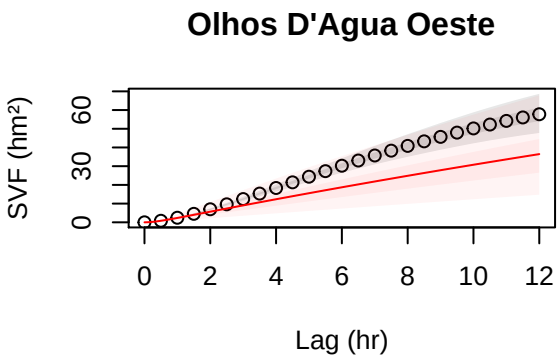
```
## [1] "individual = 2"
```

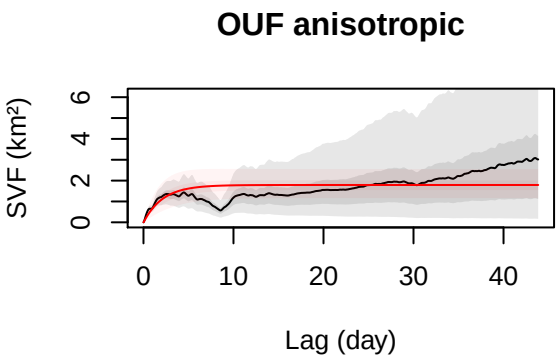


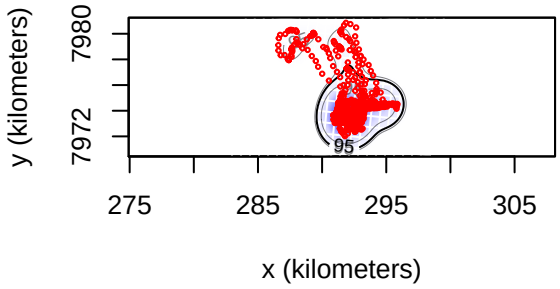




```
## [1] "individual = 3"
```

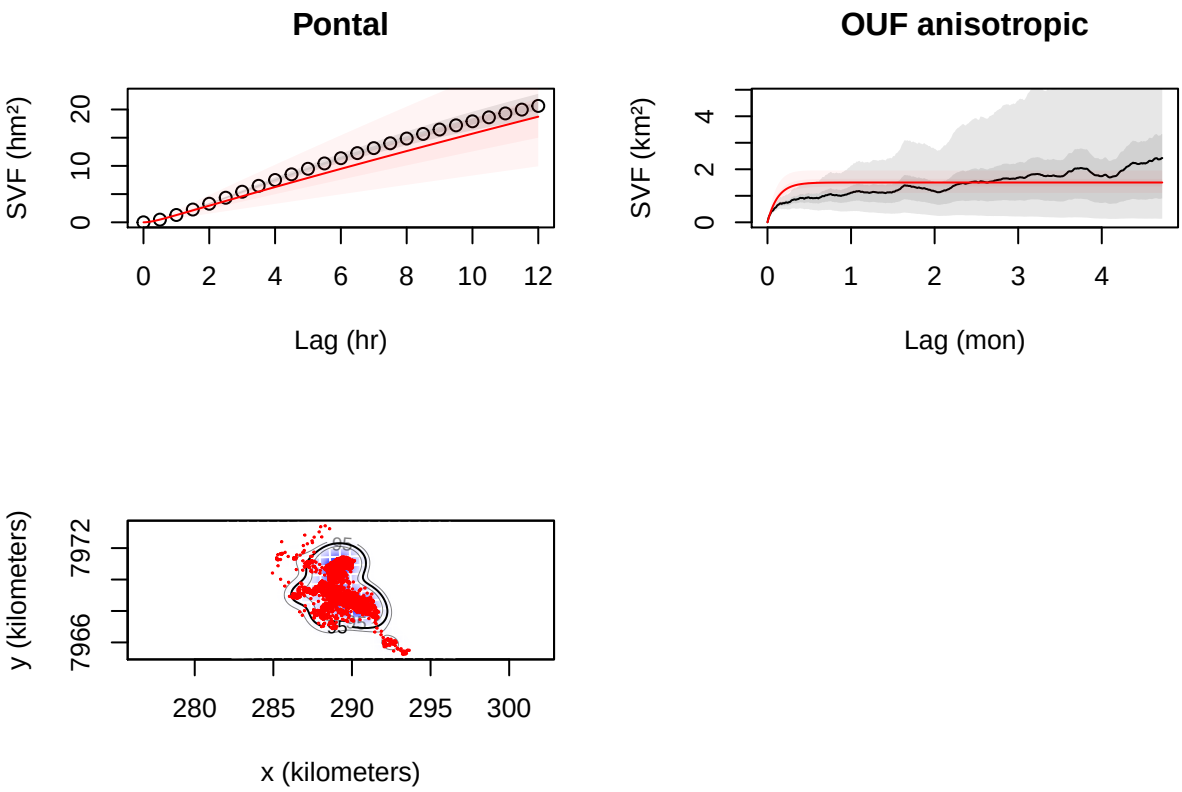



```
## [1] "individual = 4"
```

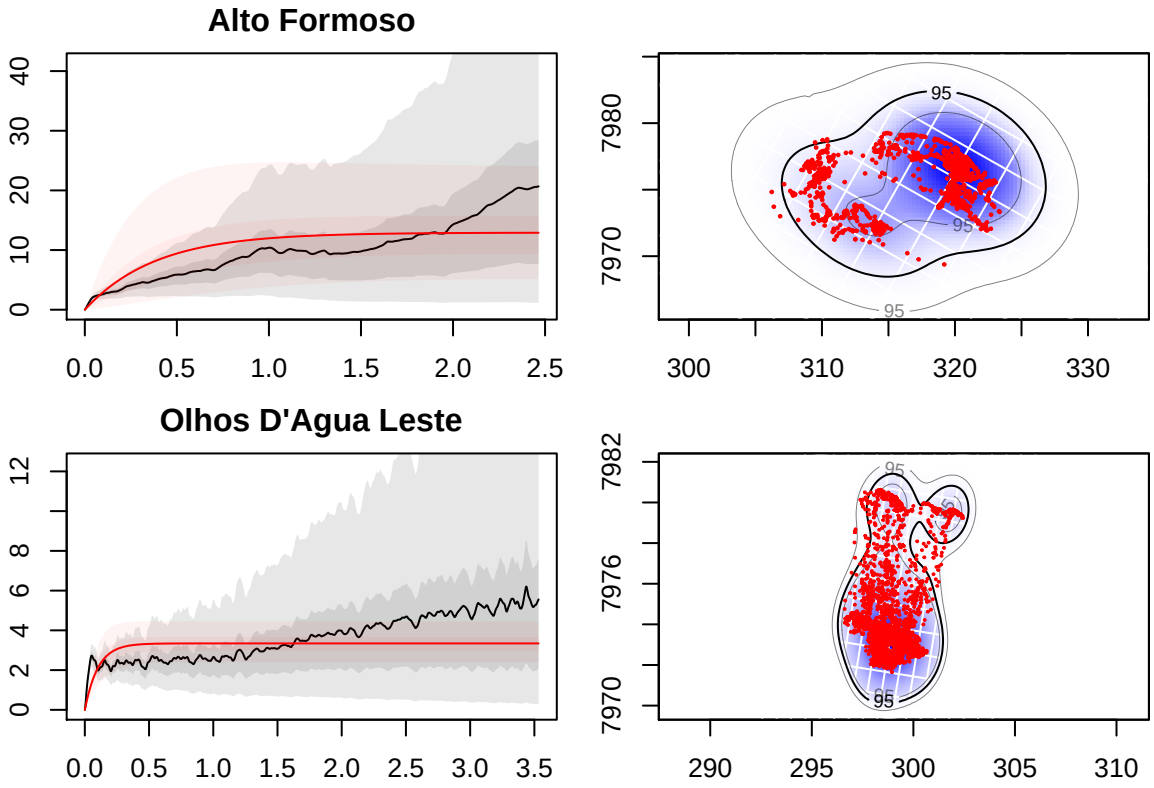
```
par(mfrow = c(2,2), mar = c(2, 2, 2, 1) + 0.1, oma = c(2,2,0,0))
```




```
for(i in 1:length(mov.tel)) {
  print(paste('individual', i, sep = ' = '))

  ind <- mov.tel[[i]]
  SVF <- variogram(ind)

  level <- c(0.5, 0.95) # 50% and 95% CIs
  xlim <- c(0, 12 %#% "hour") # 0-12 hour window

  # par(mfrow = c(2,2))
  # plot(SVF, CTMM = FITS.all[[i]][[1]], xlim=xlim, level=level)
  # title(ind@info$identity)
  plot(SVF, CTMM = FITS.all[[i]][[1]], fraction = 0.75, level=level,
       xaxt = 'n', yaxt = 'n')
  axis(1); axis(2)
  title(ind@info$identity)
  # title(names(FITS[1]))

  plot(ind, UD = AKDE.all[[i]])
  # title(rownames(summary(FITS.all[[i]]))[1])

  # krig <- occurrence(ind, CTMM = FITS[[1]])
  # plot(krig)
}
```

```
## [1] "individual = 1"
```

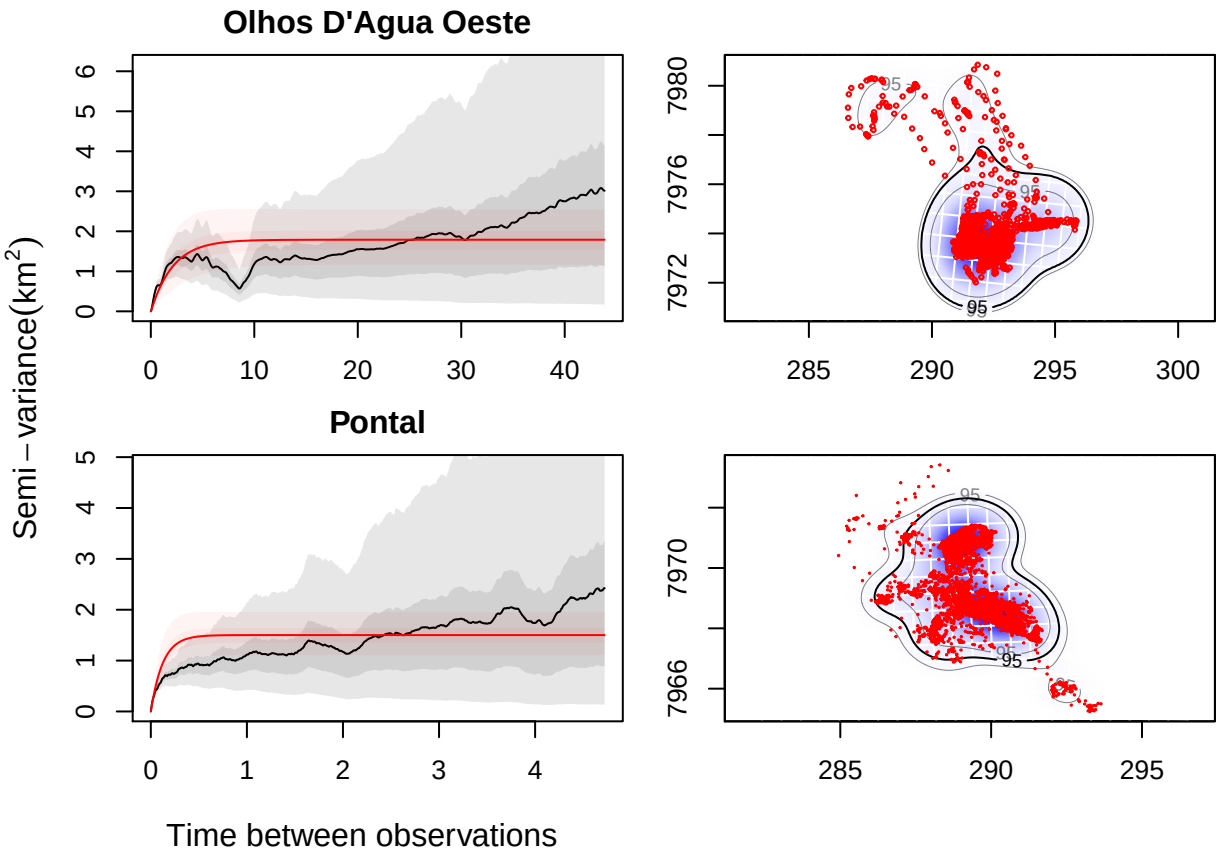
```
## [1] "individual = 2"
```




```
## [1] "individual = 3"
```

```
## [1] "individual = 4"
```

```
mtext(expression(Semi-variance (km^2)), side = 2, line = 0.5, outer = T)
mtext('Time between observations', side = 1, line = 1, outer = T,
      at = 0.25)
```




```
# write 99% akde isopleths as shapefiles
if(!dir.exists('maps/akde_99')) dir.create('maps/akde_99')
writeShapefile(AKDE.all[[1]], folder = "maps/akde_99", file = "polygontayassu_altoformoso_akde99", level
writeShapefile(AKDE.all[[2]], folder = "maps/akde_99", file = "polygontayassu_olhosleste_akde99", level
writeShapefile(AKDE.all[[3]], folder = "maps/akde_99", file = "polygontayassu_olhosoeste_akde99", level
writeShapefile(AKDE.all[[4]], folder = "maps/akde_99", file = "polygontayassu_pontal_akde99", level.UD =
```