Step selection simulation validation

Compare simulated trajectories to the observed buffalo trajectories using summary statistics. The first summary statistics are calculated for each hour of the day (i.e. mean step lengths, mean values of covariates), and the latter set summarises the full trajectory.

Scott Forrest

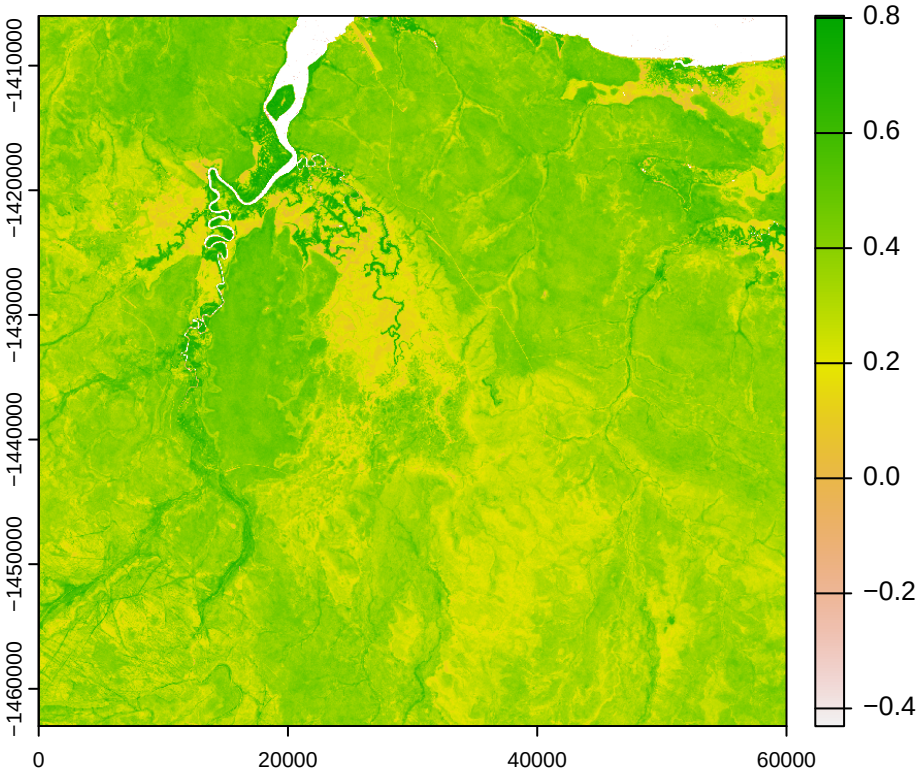
2024-02-29

Load packages

```
options(scipen=999)

library(tidyverse)
packages <- c("amt", "lubridate", "terra", "tictoc", "beepr", "ks",
              "adehabitatLT", "adehabitatHR", "ggpubr", "patchwork")
walk(packages, require, character.only = T)
```

Import the covariate raster layers that were used in the step selection model fitting.

```
# unscaled rasters
ndvi_stack <- rast("mapping/cropped rasters/ndvi_GEE_projected_watermask20230207.tif")
ndvi_2018_dry <- ndvi_stack[[8:10]]
ndvi_2019_dry <- ndvi_stack[[19:21]]
ndvi_dry <- terra::mean(c(ndvi_2018_dry, ndvi_2019_dry))
names(ndvi_dry) <- "ndvi_dry"
plot(ndvi_dry)
```




```
canopy <- rast("mapping/cropped rasters/canopy_cover.tif")
herby <- rast("mapping/cropped rasters/veg_herby.tif")
slope <- rast("mapping/cropped rasters/slope_raster.tif")
```

Load simulated trajectories and extract covariate information

In this script we are only calculating the summaries for the simulations of a single model. To compare the simulations of the other models, we would just change the folder name that we are reading in the csv files from.

We have the trajectories saved in certain folders, so we make a list of all the files in that folder, and then filter them based on the trajectories that we want to select (using the `grep` function to match the strings). Each file will have many trajectories for the starting locations of each of the buffalo (depending on how the simulation was set up). We then read in the csv files.

The only change between the simulations for different IDs in this case is the starting location, but it's also possible to use different memory parameters or coefficients if the models are fitted to each individual animal.

```
# read in multiple csv files with similar filenames and bind them together
sim_data_full_list <-
  list.files("outputs/simulated trajectories/CLR TS 3pDaily GvM MemNatALL/",
             pattern = "*.csv", full.names = T)

sim_data_all <- grep("50ch", sim_data_full_list, value = T) %>%
  grep("3000", x = ., value = T) %>%
  map_dfr(read_csv, .id = "file_num") %>%
  mutate(file_num = str_remove(file_num, ".csv")) %>%
```

```
  mutate(file_num = as.numeric(file_num))

# to check the initial locations of the simulated trajectories -
# prior to removing the warmup period - these should be the same as the observed
# data
sim_data_all %>% group_by(id) %>% slice(1)
```

```
## # A tibble: 520 x 10
## # Groups:   id [520]
##    file_num id           x_        y_ t_                  angle bearing hab_p mem_p  hour
##       <dbl> <chr>     <dbl>     <dbl> <dttm>               <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1        1 id20051  41968. -1435673. 2019-08-01 01:00:00  2.32        0     0     0     1
##  2        1 id200510 41968. -1435673. 2019-08-01 01:00:00 -1.80        0     0     0     1
##  3        2 id200511 41968. -1435673. 2019-08-01 01:00:00  0.559       0     0     0     1
##  4        2 id200512 41968. -1435673. 2019-08-01 01:00:00  2.09        0     0     0     1
##  5        2 id200513 41968. -1435673. 2019-08-01 01:00:00  2.98        0     0     0     1
##  6        2 id200514 41968. -1435673. 2019-08-01 01:00:00  0.480       0     0     0     1
##  7        2 id200515 41968. -1435673. 2019-08-01 01:00:00  0.927       0     0     0     1
##  8        2 id200516 41968. -1435673. 2019-08-01 01:00:00  0.309       0     0     0     1
##  9        2 id200517 41968. -1435673. 2019-08-01 01:00:00  0.223       0     0     0     1
## 10        2 id200518 41968. -1435673. 2019-08-01 01:00:00 -0.505       0     0     0     1
## # i 510 more rows
```

Prepare the simulated data

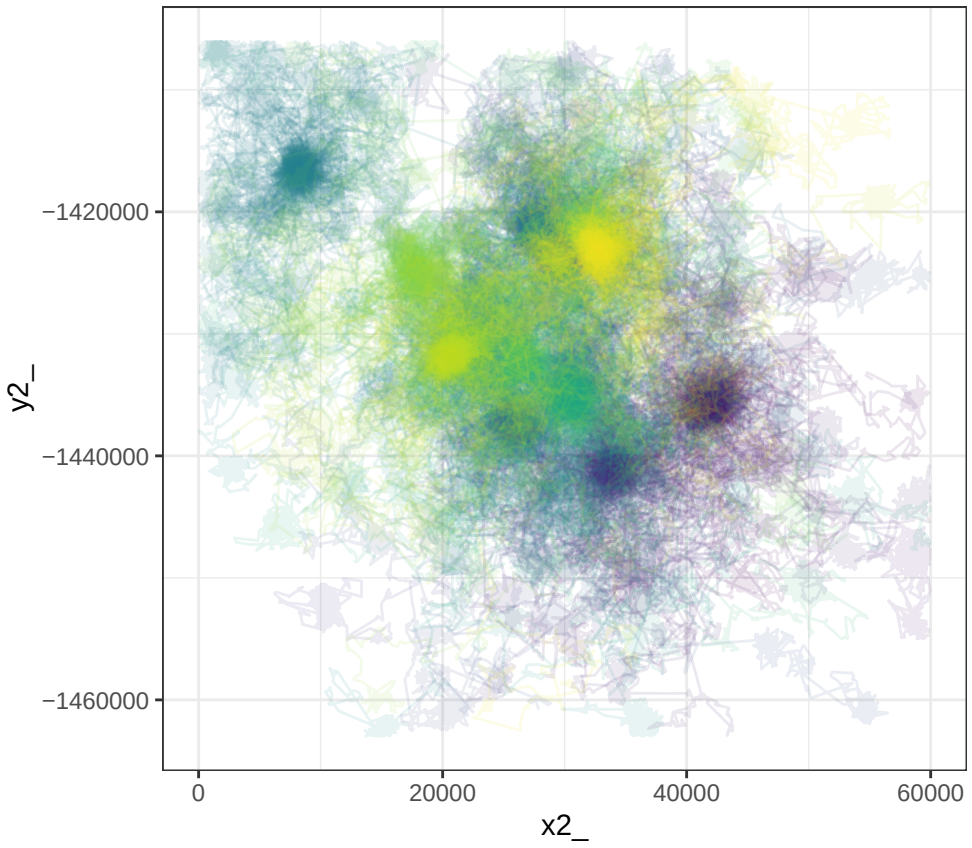
We need to do a bit of tidying, mostly just so we have a unique ID for each of the simulated trajectories, and then we convert to a track object using the `amt` package and extract covariate values.

```
# parse out the id column string to have only the four digit numeric id,
# and a separate column which is the simulation number (the 5th numeric digit)
sim_data_all <- sim_data_all %>% mutate(id = str_remove(id, "id"),
                                        hour = ifelse(hour == 0, 24, hour))

# take the 5th digit of the id column and make it a separate column,
# but where the sim_number is double digits, take the 5th and the 6th digit
sim_data_all <- sim_data_all %>% mutate(sim_num = str_sub(id, 5, 7))
str_sub(sim_data_all$id, 5, 7) <- ""

# create a new column which is combines id, sim_number and file_number
# as these represent different trajectories
sim_data_all <- sim_data_all %>% mutate(traj = paste(id, sim_num, file_num, sep = "_"))

# rearrange the columns so it is id num, sim num, file num then the rest
sim_data_all <- sim_data_all %>% dplyr::select(id, file_num, sim_num, traj, everything())

# create vector of unique ids for subsetting
sim_data_traj_ids <- unique(sim_data_all$traj)

# attach ACST time format
tz(sim_data_all$t_) <- "Australia/Darwin"
# subtract a year from the time column so it matches the buffalo data
# minus the warmup period (500 hours)
sim_data_all$t_ <- sim_data_all$t_ - years(1) - days(31)
```

```r
# convert to track object for step lengths and turning angles etc
sim_data_all <- sim_data_all %>% mk_track(id = id, x_, y_, t_, all_cols = T, crs = 3112)
sim_data_all_nested <- sim_data_all %>% arrange(traj) %>% nest(data = -"traj")

# extract covariate values at the end of the all the simulated steps
sim_data_all_nested_steps <- sim_data_all_nested %>%

  mutate(steps = map(data, function(x)
    x %>% steps(keep_cols = "end") %>%
      extract_covariates(ndvi_dry,
                         where = "end") %>%
      extract_covariates(canopy,
                         where = "end") %>%
      extract_covariates(herby,
                         where = "end") %>%
      extract_covariates(slope,
                         where = "end")))

# sim_data_all_nested_steps$steps[[1]]
sim_data_all_steps <- sim_data_all_nested_steps %>%
  amt::select(traj, steps) %>%
  amt::unnest(cols = steps)

# rearrange the columns so it is id num, sim num, file num then the rest
sim_data_all_steps <- sim_data_all_steps %>%
  dplyr::select(id, file_num, sim_num, traj, everything()) %>%
  mutate(month = month(t1_))

sim_data_all_track <- sim_data_all_steps %>%
  mk_track(id = id, x1_, y1_, t1_, order_by_ts = T, all_cols = T, crs = 3112) %>%
  arrange(traj)

sim_ids <- unique(sim_data_all_steps$id)
```

Plot all simulated individuals spatially

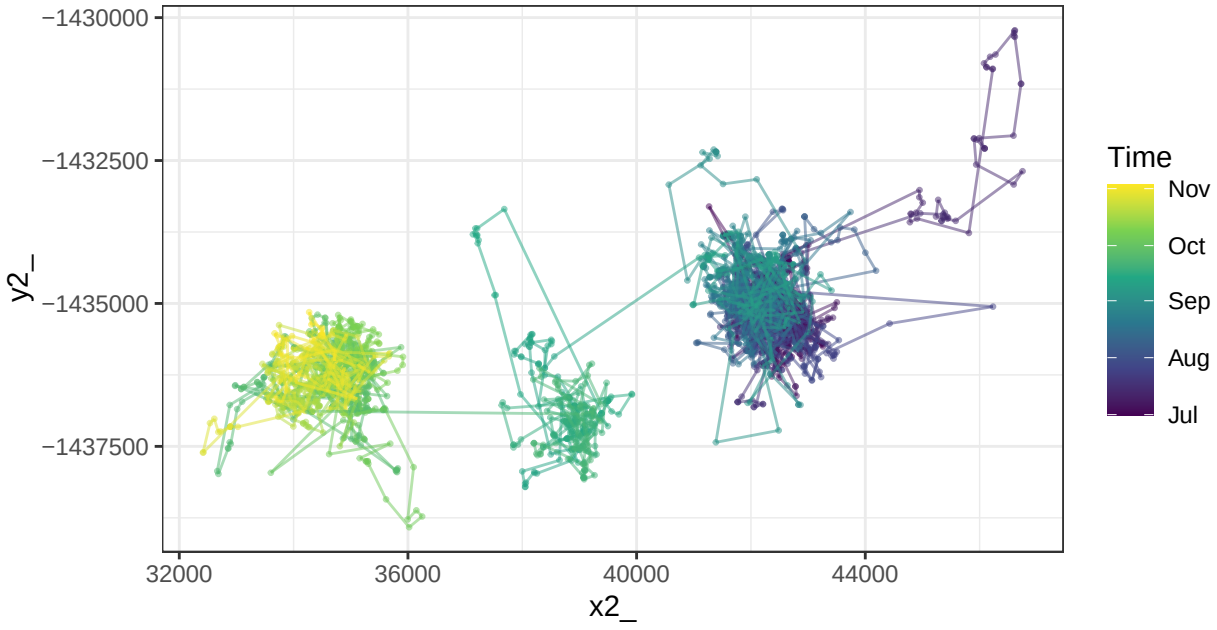
```r
ggplot() +
  geom_path(data = sim_data_all_steps,
            aes(x = x2_, y = y2_, colour = traj),
            alpha = 0.1) +
  scale_color_viridis_d("Sim ID") +
  coord_equal() +
  theme_bw() +
  theme(legend.position = "none")
```

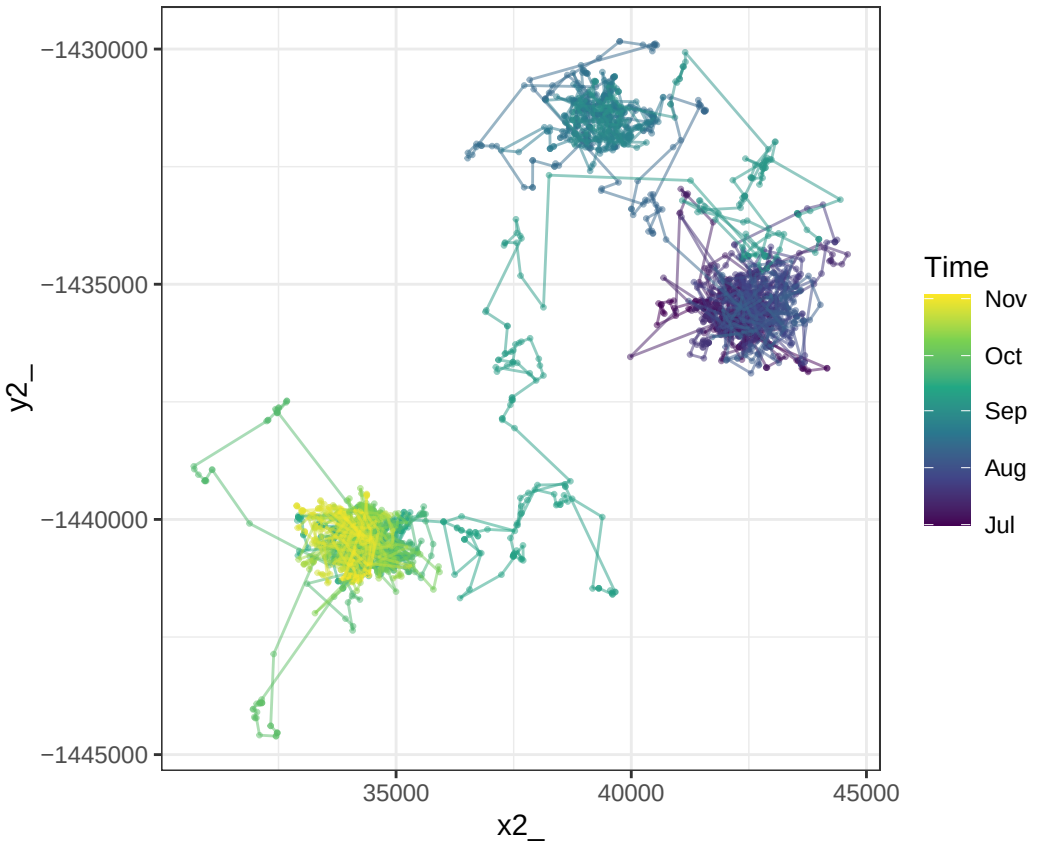


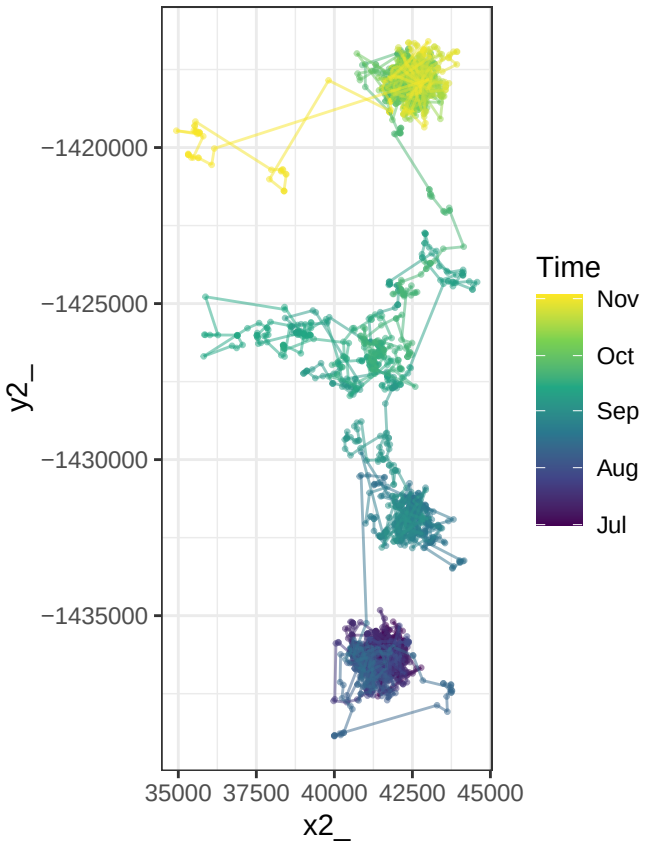
Plot separate individuals coloured by time

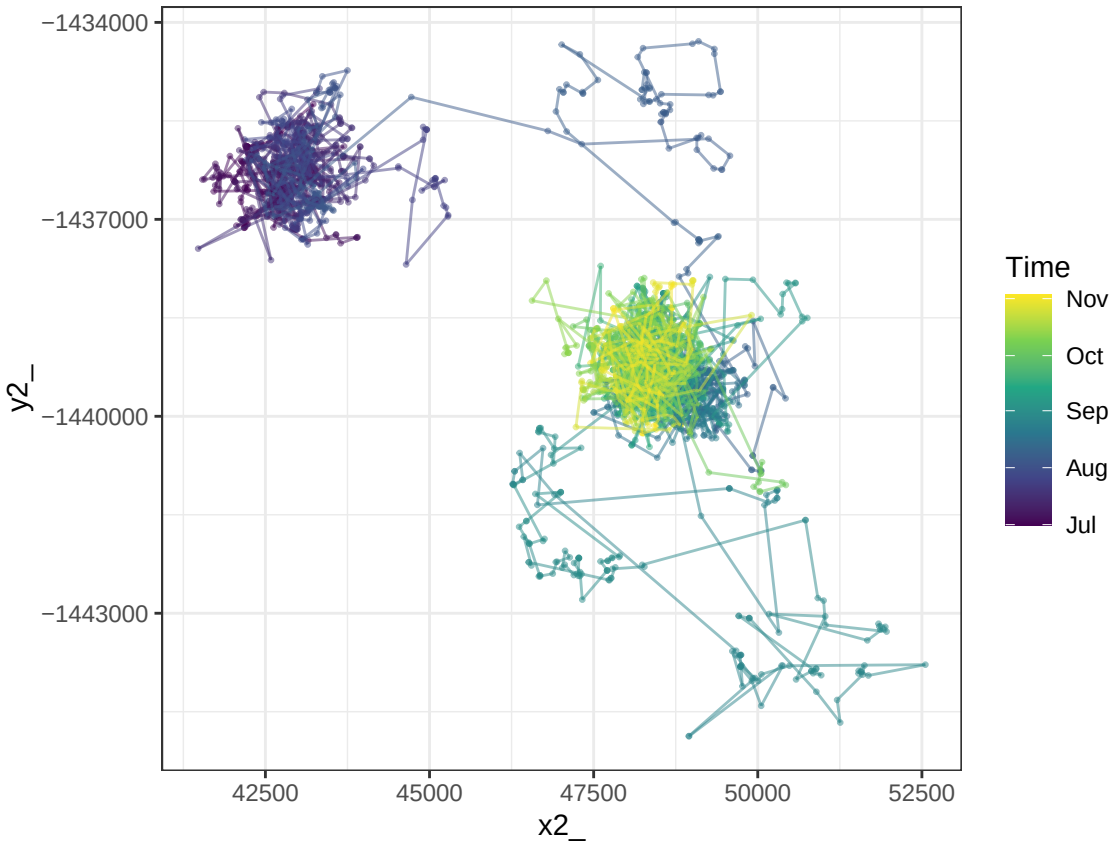
Just plot the first 5 individuals as an example.

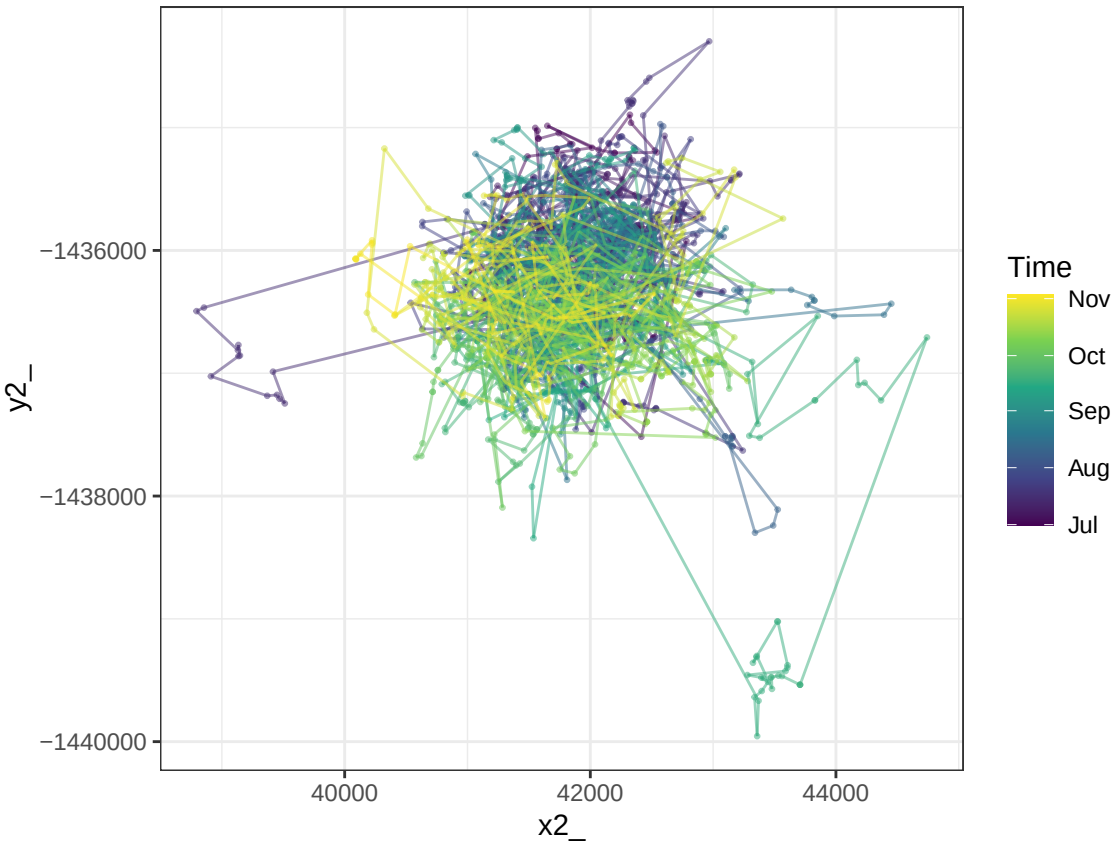
```
for(i in 1:5) { # length(sim_data_ids)
 print(ggplot() +
         geom_point(data = sim_data_all_steps %>%
                      filter(traj == sim_data_traj_ids[i]),
                    aes(x = x2_, y = y2_, colour = date(t2_)),
                    size = 0.5, alpha = 0.5) +
         geom_path(data = sim_data_all_steps %>%
                     filter(traj == sim_data_traj_ids[i]),
                   aes(x = x2_, y = y2_, colour = date(t2_)),
                   alpha = 0.5) +
         scale_color_viridis_c("Time", trans = "date") +
         coord_equal() +
         theme_bw())
}
```


−1430000
−1432500
−1435000
−1437500
y2_
32000
36000
40000
44000
x2_
Time
Nov
Oct
Sep
Aug
Jul


−1430000
−1435000
−1440000
−1445000
y2_
35000
40000
45000
x2_
Time
Nov
Oct
Sep
Aug
Jul


−1420000
−1425000
−1430000
−1435000
y2_
35000
37500
40000
42500
45000
x2_
Time
Nov
Oct
Sep
Aug
Jul


−1434000
−1437000
−1440000
−1443000
y2_
42500
45000
47500
50000
52500
x2_
Time
Nov
Oct
Sep
Aug
Jul



Load observed buffalo data for comparison

```
# importing data, mostly for plotting
buffalo_data_all <-
  read_csv("outputs/buffalo_popn_GvM_covs_ST_KDEmem1000_allOPTIM_10rs_2024-02-05.csv")
```

```
## Rows: 1082697 Columns: 26
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (22): id, burst_, x1_, x2_, y1_, y2_, sl_, ta_, dt_, hour_t2, step_id_, y, ndvi_temporal, veg_w
## lgl   (1): case_
## dttm  (3): t1_, t2_, t2_rounded
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
buffalo_data_all <- buffalo_data_all %>%
  mutate(t1_ = lubridate::with_tz(buffalo_data_all$t1_, tzone = "Australia/Darwin"),
         t2_ = lubridate::with_tz(buffalo_data_all$t2_, tzone = "Australia/Darwin"))

buffalo_data_all <- buffalo_data_all %>%
  mutate(id_num = as.numeric(factor(id)),
         step_id = step_id_,
         x1 = x1_, x2 = x2_,
         y1 = y1_, y2 = y2_,
```

```
         t1 = t1_,
         t1_rounded = round_date(t1_, "hour"),
         hour_t1 = hour(t1_rounded),
         t2 = t2_,
         t2_rounded = round_date(t2_, "hour"),
         hour_t2 = hour(t2_rounded),
         hour = ifelse(hour_t2 == 0, 24, hour_t2),
         yday = yday(t1_),
         year = year(t1_),
         month = month(t1_),
         sl = sl_,
         log_sl = log(sl_),
         ta = ta_,
         cos_ta = cos(ta_),
         canopy_01 = canopy_cover/100,
         spatiotemporal_memory_density_log = kde_memory_density_log,
         spatiotemporal_memory_density = exp(kde_memory_density_log))

# ensure that we have just the individuals that were used for model fitting
buffalo_year_ids <- c(2005, 2014, 2018, 2021, 2022, 2024, 2039,
                      2154, 2158, 2223, 2327, 2387, 2393)
buffalo_data_all <- buffalo_data_all %>% filter(id %in% buffalo_year_ids)
buffalo_data_all <- buffalo_data_all %>% filter(t1 < "2019-07-25 09:32:42 ACST")
buffalo_ids <- unique(buffalo_data_all$id)

# check the initial location and time for each individual
buffalo_data_all %>% group_by(id) %>% slice(1)
```

```
## # A tibble: 13 x 47
## # Groups:   id [13]
##       id burst_    x1_    x2_       y1_       y2_     sl_   ta_ t1_                 t2_
##    <dbl>  <dbl>  <dbl>  <dbl>     <dbl>     <dbl>   <dbl> <dbl> <dttm>              <dttm>
##  1  2005      1 41968. 41921. -1435673. -1435656.   50.7   1.37 2018-07-25 10:34:23 2018-07-25 11:34
##  2  2014      1 33615. 33611. -1440916. -1440912.    5.58 NA    2018-07-25 09:32:42 2018-07-25 10:32
##  3  2018      2 30093. 30097. -1423073. -1423080.    7.94 NA    2018-07-25 12:25:36 2018-07-25 13:26
##  4  2021      1 25964. 26059. -1437033. -1437137.  140.   NA    2018-07-25 10:07:02 2018-07-25 11:08
##  5  2022      1 27928. 27930. -1421152. -1421158.    5.95 NA    2018-07-25 10:05:16 2018-07-25 11:06
##  6  2024      1  8227.  8010. -1416224. -1416494.  346.   NA    2018-07-25 10:02:44 2018-07-25 11:02
##  7  2039      1 21574. 20933. -1431773. -1431738.  642.   NA    2018-07-25 09:40:05 2018-07-25 10:39
##  8  2154      1 30551. 30550. -1434891. -1434892.    1.43 NA    2018-07-25 10:14:19 2018-07-25 11:14
##  9  2158      1 26817. 26606. -1432574. -1433024.  498.   NA    2018-07-25 10:11:00 2018-07-25 11:10
## 10  2223      1 31819. 34054. -1421650. -1421202. 2279.   NA    2018-07-25 10:17:28 2018-07-25 11:16
## 11  2327      1 17867. 19300. -1424144. -1425091. 1718.   NA    2018-07-25 09:44:01 2018-07-25 10:45
## 12  2387      1 25451. 23473. -1431346. -1431678. 2006.   NA    2018-07-25 09:39:57 2018-07-25 10:40
## 13  2393      2 32549. 32550. -1423007. -1423011.    4.35 NA    2018-07-25 11:41:31 2018-07-25 12:40
## # i 36 more variables: t2_rounded <dttm>, hour_t2 <int>, case_ <lgl>, step_id_ <dbl>, y <dbl>, ndvi_
## #   veg_woody <dbl>, veg_herby <dbl>, canopy_cover <dbl>, DEM_H <dbl>, slope <dbl>, WOFS <dbl>, cos_
## #   log_sl_ <dbl>, kde_memory_density_log <dbl>, id_num <dbl>, step_id <dbl>, x1 <dbl>, x2 <dbl>, y1
## #   t1 <dttm>, t1_rounded <dttm>, hour_t1 <int>, t2 <dttm>, hour <dbl>, yday <dbl>, year <dbl>, montl
## #   log_sl <dbl>, ta <dbl>, cos_ta <dbl>, canopy_01 <dbl>, spatiotemporal_memory_density_log <dbl>,
## #   spatiotemporal_memory_density <dbl>
```

```
# make a dataframe of only the presence locations
buffalo_data_all_pres <- buffalo_data_all %>% filter(y == 1)
```

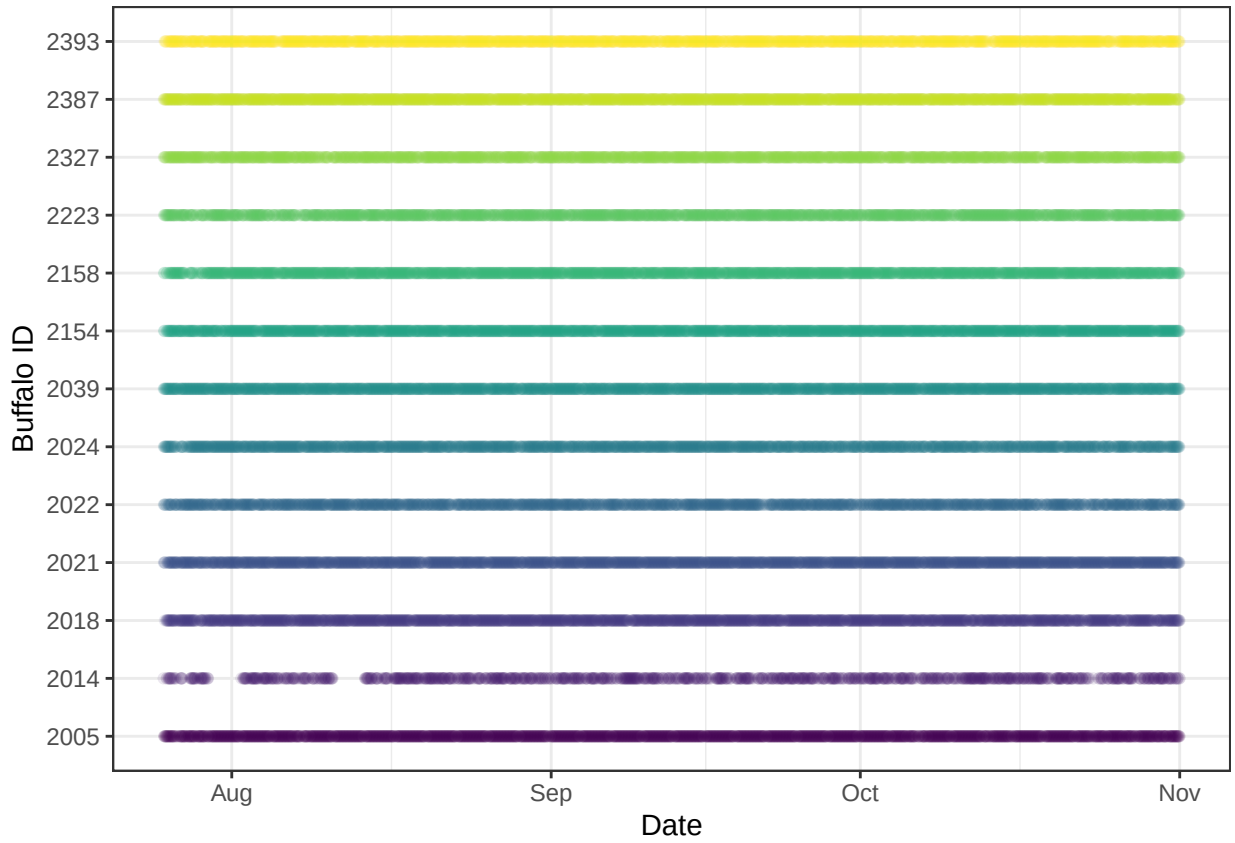
```
# convert to track object for step lengths and turning angles etc
buffalo_data_all_pres <- buffalo_data_all_pres %>%
  mk_track(id = id, x1_, y1_, t1_, order_by_ts = T, all_cols = T, crs = 3112) %>%
  arrange(id) %>%
  extract_covariates(ndvi_dry, where = "end")
```

Subset the buffalo data to the period that simulations were generated for

```
months_wet <- c(1:4, 11:12)

buffalo_data <- buffalo_data_all_pres %>%
  filter(!month %in% months_wet & year == 2018) # dry season 2018

buffalo_data_nested <- buffalo_data %>% arrange(id) %>% nest(data = -"id")
```

Plot timeline of GPS data

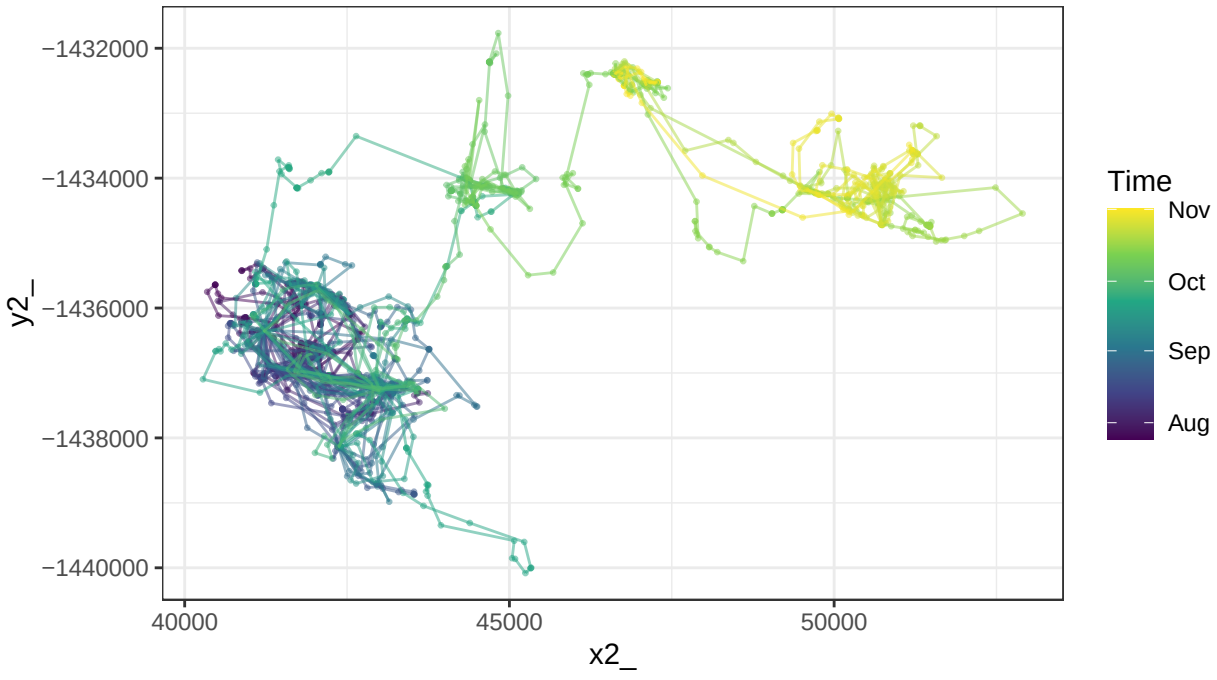
```
buffalo_data %>% ggplot(aes(x = t1, y = factor(id), colour = factor(id))) +
  geom_point(alpha = 0.1) +
  scale_y_discrete("Buffalo ID") +
  scale_x_datetime("Date") +
  scale_colour_viridis_d() +
  theme_bw() +
  theme(legend.position = "none")
```

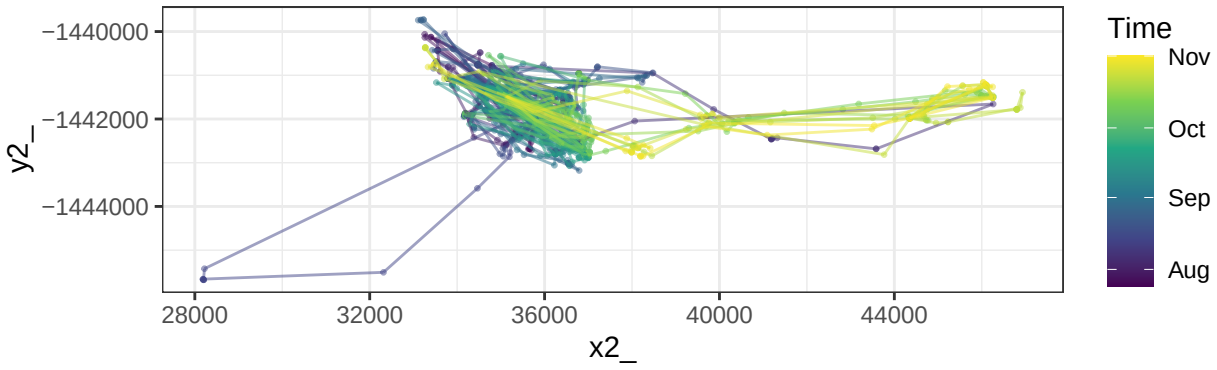


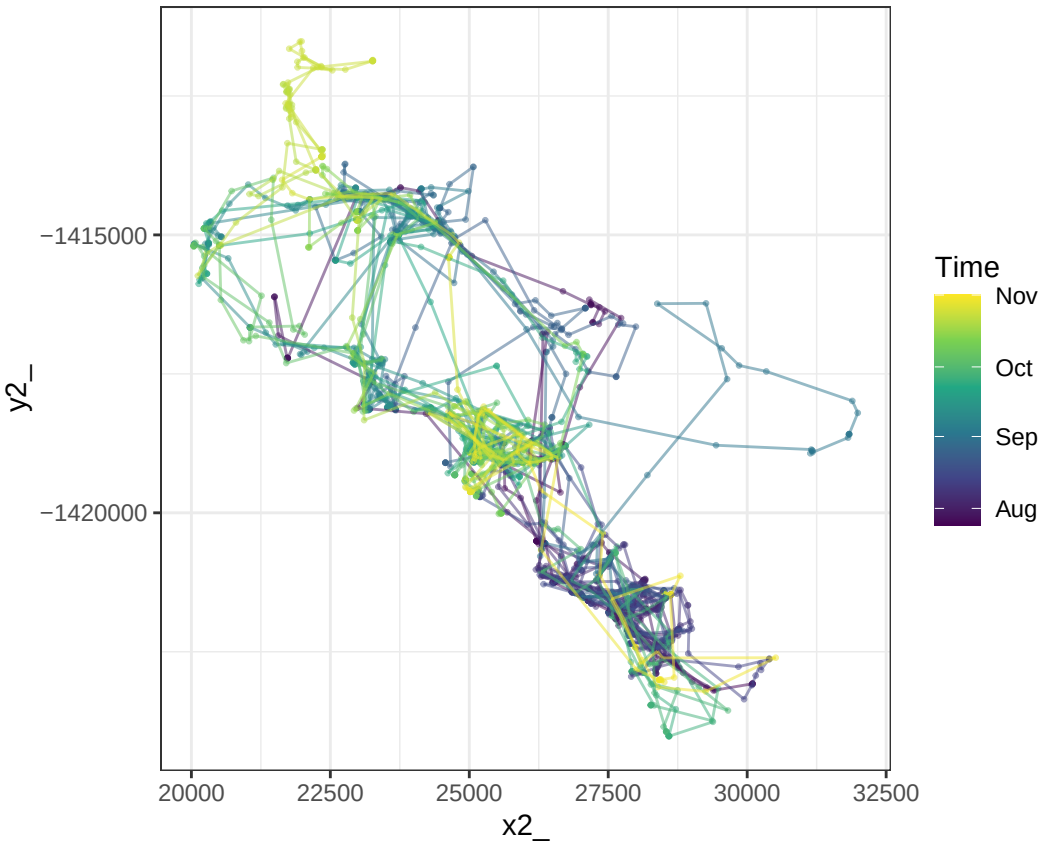
Plot each buffalo's data coloured by time

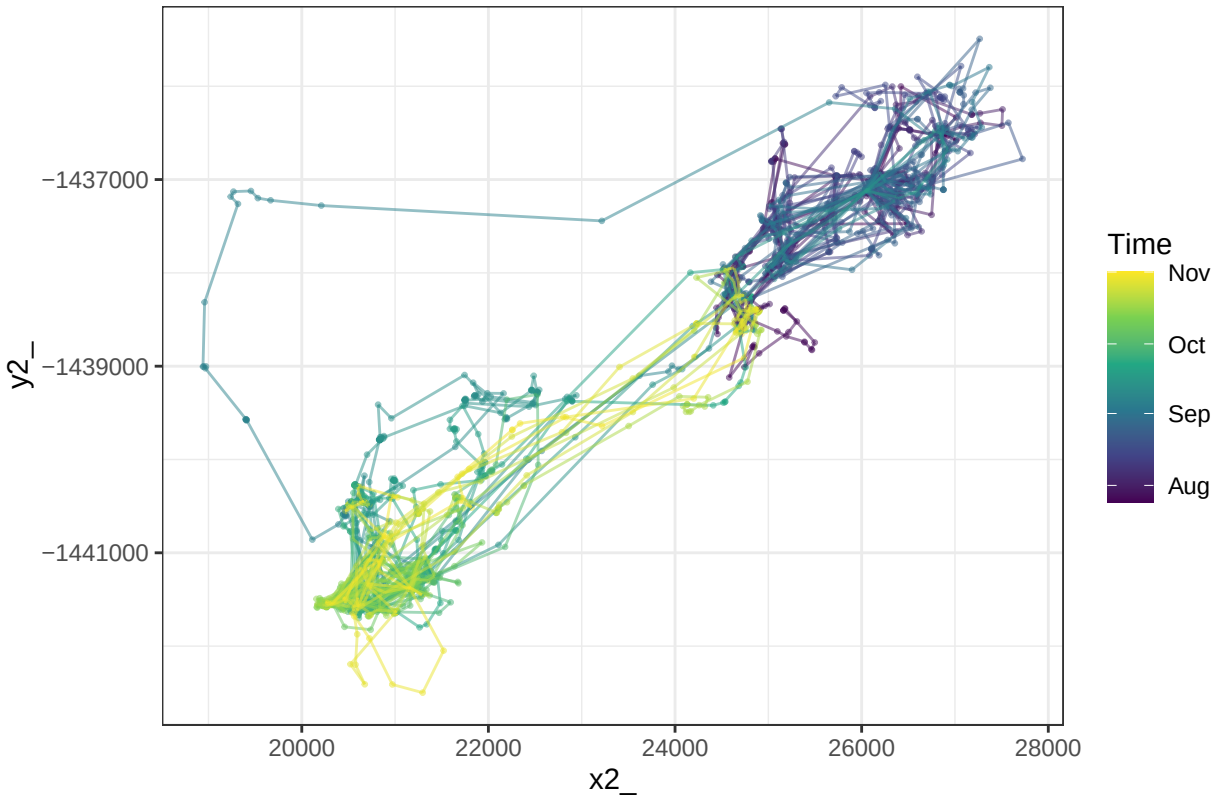
It's clear that there is a structure to the buffalo data that is not replicated with the simulated trajectories. Incorporating summary statistics that can discern between the observed and simulated data would be useful.

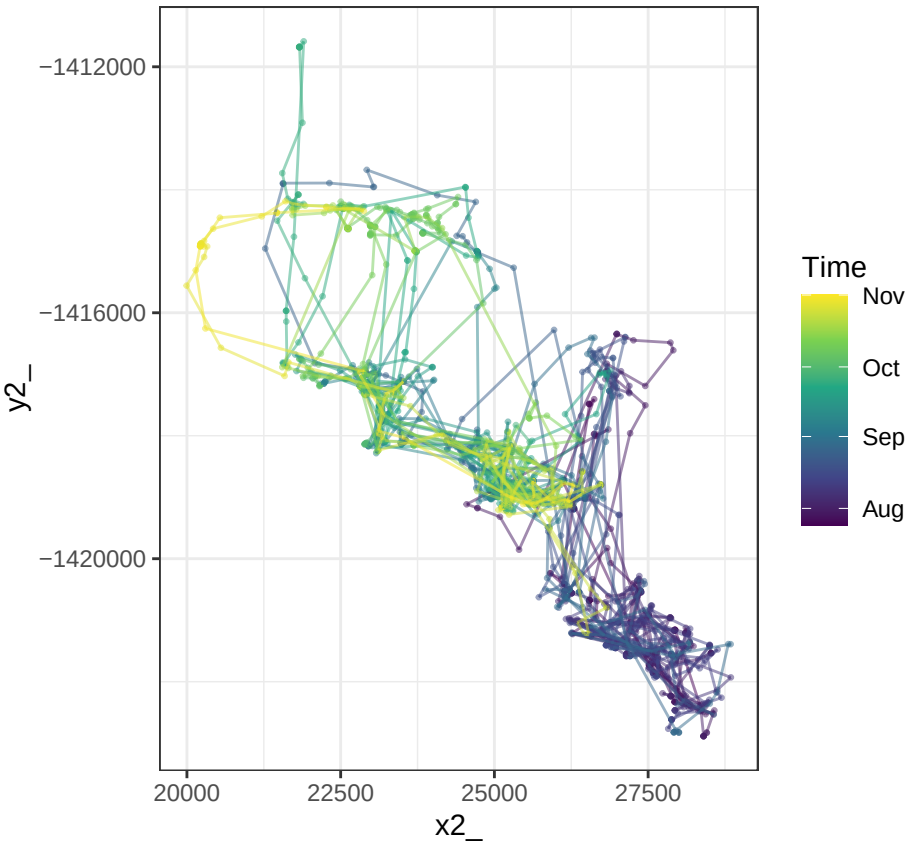
```
for(i in 1:length(unique(buffalo_data$id))) {
 print(ggplot() +
    geom_point(data = buffalo_data %>% filter(id == unique(buffalo_data$id)[i]),
               aes(x = x2_, y = y2_, colour = date(t2_)),
               size = 0.5, alpha = 0.5) +
    geom_path(data = buffalo_data %>% filter(id == unique(buffalo_data$id)[i]),
              aes(x = x2_, y = y2_, colour = date(t2_)),
              alpha = 0.5) +
    scale_color_viridis_c("Time", trans = "date") +
    coord_equal() +
    theme_bw())
}
```

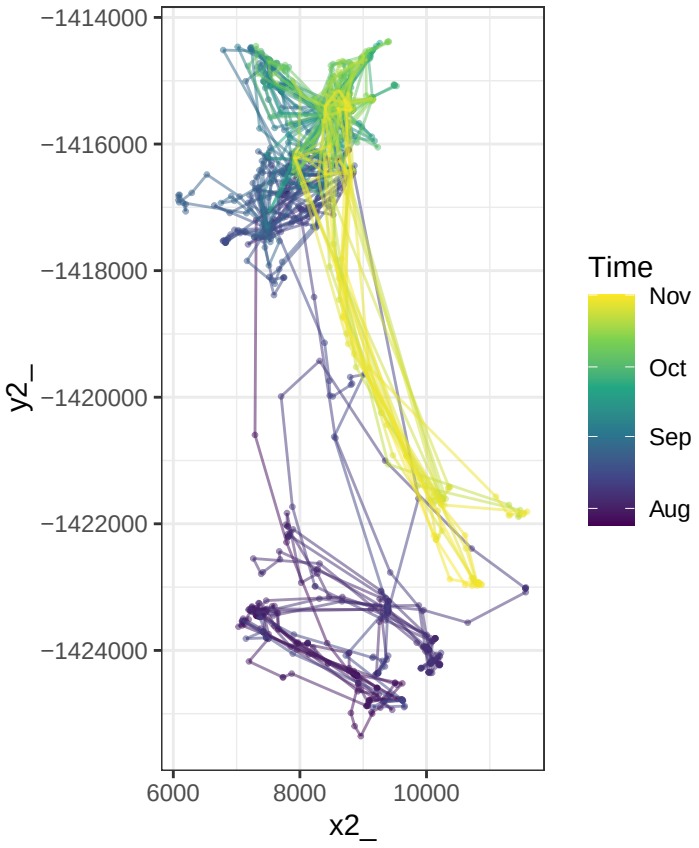


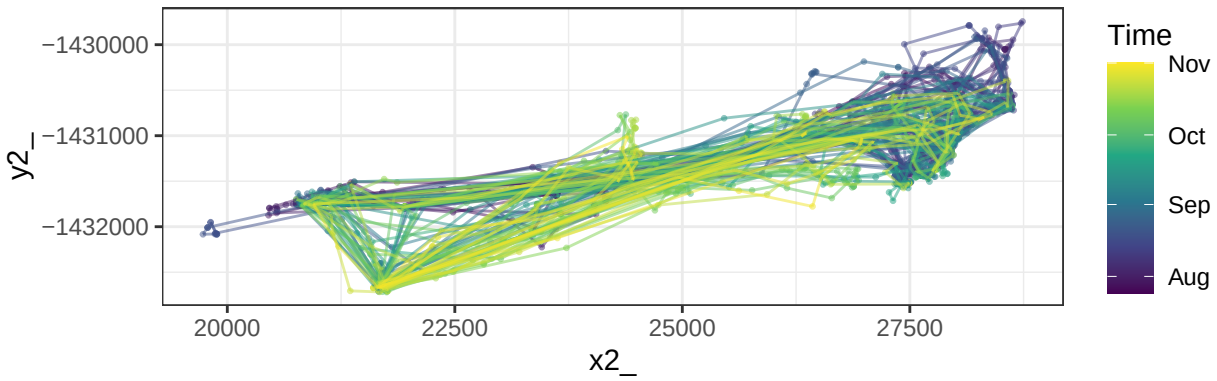

−1440000
−1442000
−1444000
y2_
28000
32000
36000
40000
44000
x2_
Time
Nov
Oct
Sep
Aug

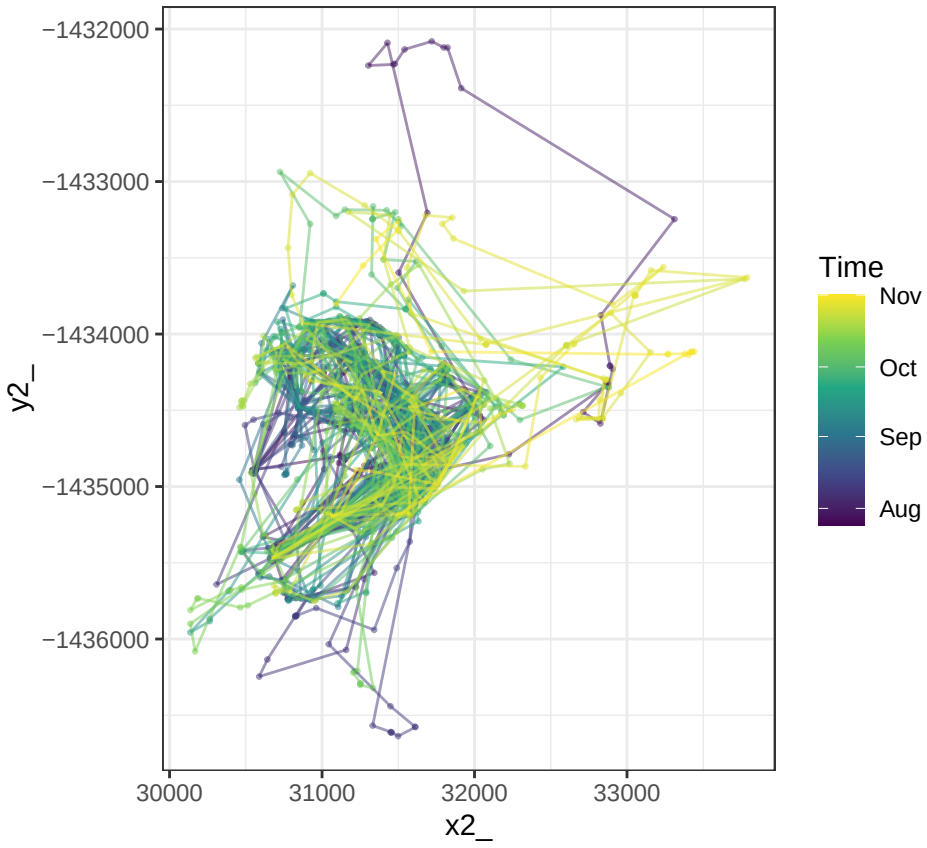

−1415000
−1420000
y2_
20000
22500
25000
27500
30000
32500
x2_
Time
Nov
Oct
Sep
Aug

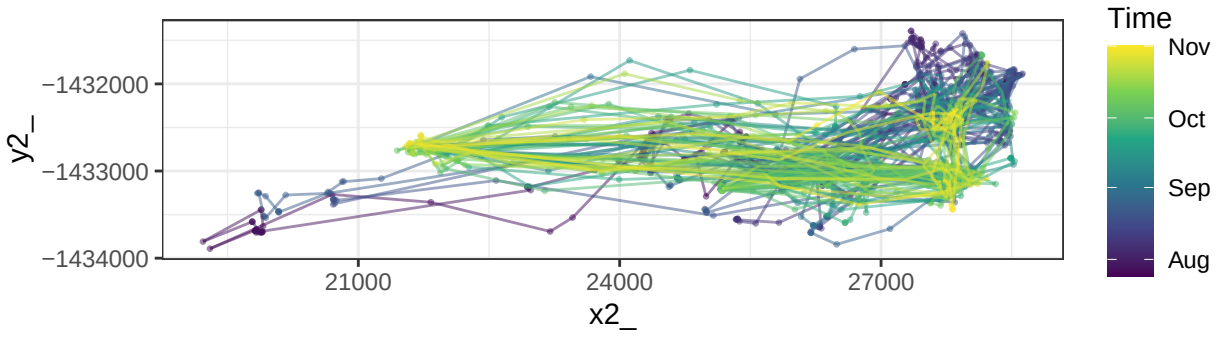

−1437000
−1439000
−1441000
y2_
20000
22000
24000
26000
28000
x2_
Time
Nov
Oct
Sep
Aug

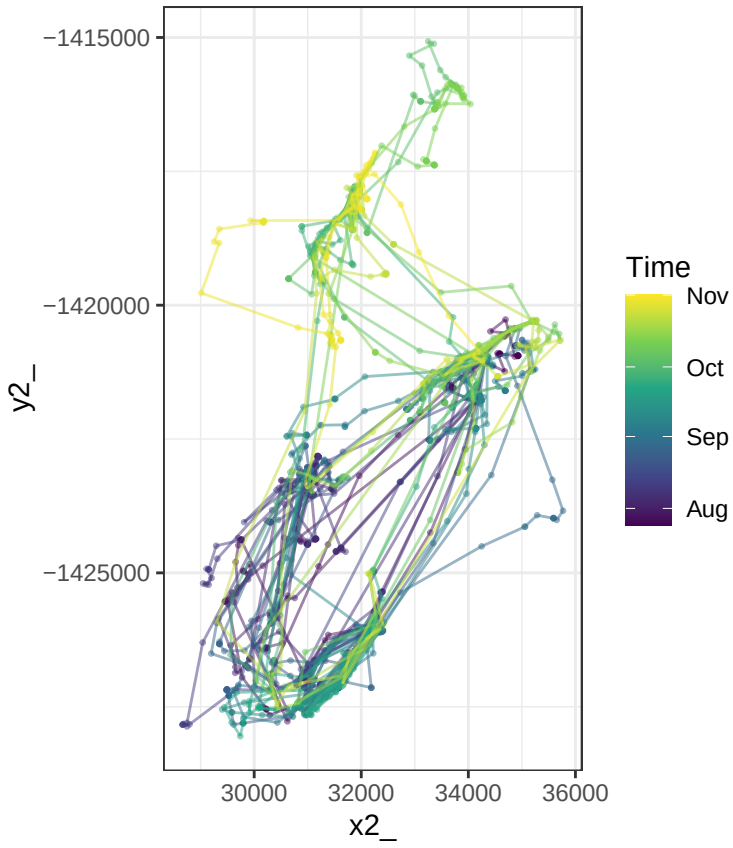

−1412000
−1416000
−1420000
y2_
20000
22500
25000
27500
x2_
Time
Nov
Oct
Sep
Aug

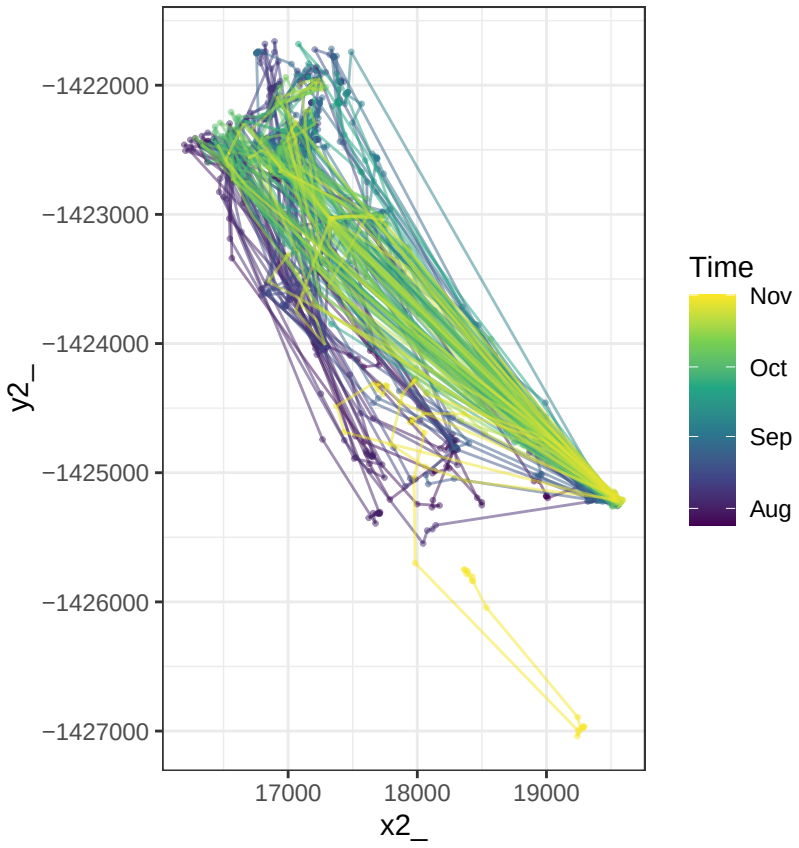

−1414000
−1416000
−1418000
−1420000
−1422000
−1424000
y2_
6000
8000
10000
x2_
Time
Nov
Oct
Sep
Aug

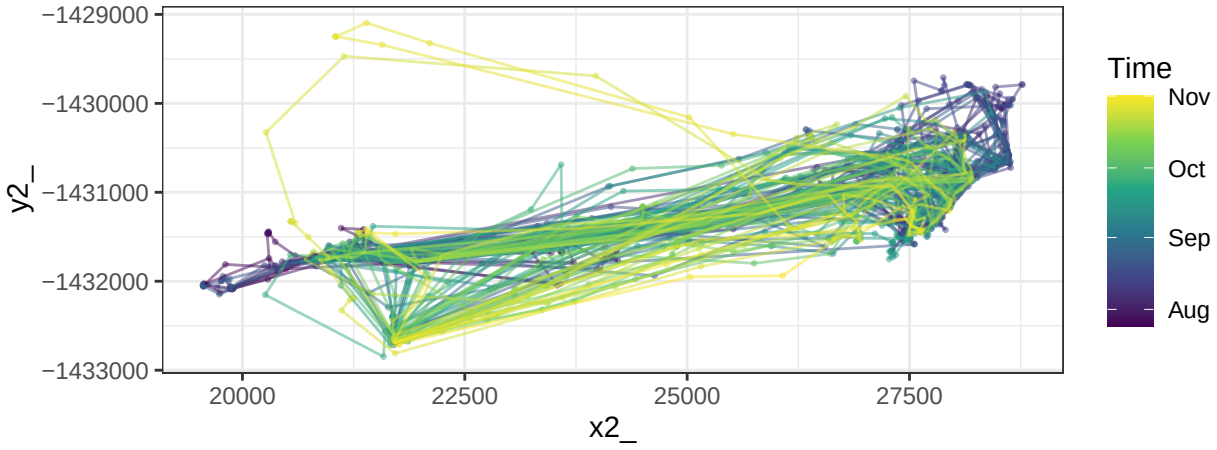

−1430000
−1431000
−1432000
y2_
20000
22500
25000
27500
x2_
Time
Nov
Oct
Sep
Aug

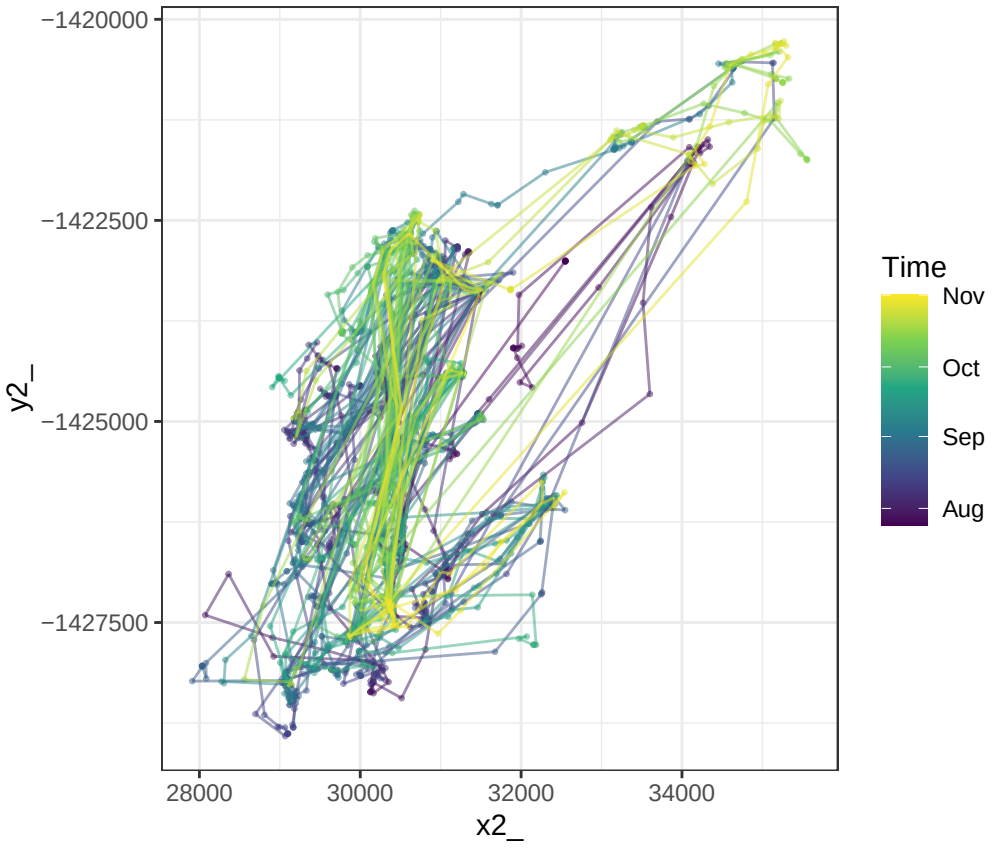

−1432000
−1433000
−1434000
−1435000
−1436000
y2_
30000
31000
32000
33000
x2_
Time
Nov
Oct
Sep
Aug


Time
Nov
Oct
Sep
Aug
−1432000
−1433000
−1434000
y2_
21000
24000
27000
x2_


−1415000
−1420000
−1425000
y2_
30000
32000
34000
36000
x2_
Time
Nov
Oct
Sep
Aug


−1422000
−1423000
−1424000
−1425000
−1426000
−1427000
y2_
17000
18000
19000
x2_
Time
Nov
Oct
Sep
Aug


−1429000
−1430000
−1431000
−1432000
−1433000
y2_
20000
22500
25000
27500
x2_
Time
Nov
Oct
Sep
Aug



Keep the simulated data that lines up with the observed data, for each individual buffalo

Here we keep just the locations that relate to the observed data, for each individual buffalo. This is to maintain any temporal biases that are present in the observed data in the simulated data.

```
# filter the simulated data to contain only the relevant time
# points for each individual buffalo
sim_data_btime_list <- vector(mode = "list", length = length(buffalo_ids))

for(j in 1:length(buffalo_ids)) {

  buffalo_data_id <- buffalo_data %>% filter(id == buffalo_ids[j])
  sim_data_id <- sim_data_all_track %>% filter(id == buffalo_ids[j])
  # retain the simulated data only when it corresponds to the same time
  # as in the buffalo data
  sim_data_btime_list[[j]] <- sim_data_id %>%
    filter(t2_ %in% buffalo_data_id$t2_rounded)

}

# combine list elements into data frame
sim_data_btime <- bind_rows(sim_data_btime_list)

sim_data_traj_ids <- unique(sim_data_btime$traj)
```

Plot each individual against simulated data

Here we show some example trajectories plotted against the observed data for that starting location. To make a better comparison for plotting, we choose simulations with starting locations that are near to the buffalo starting locations. As we discarded the memory warm-up period (about 500 locations), the actual starting location of the simulated data typically differs.

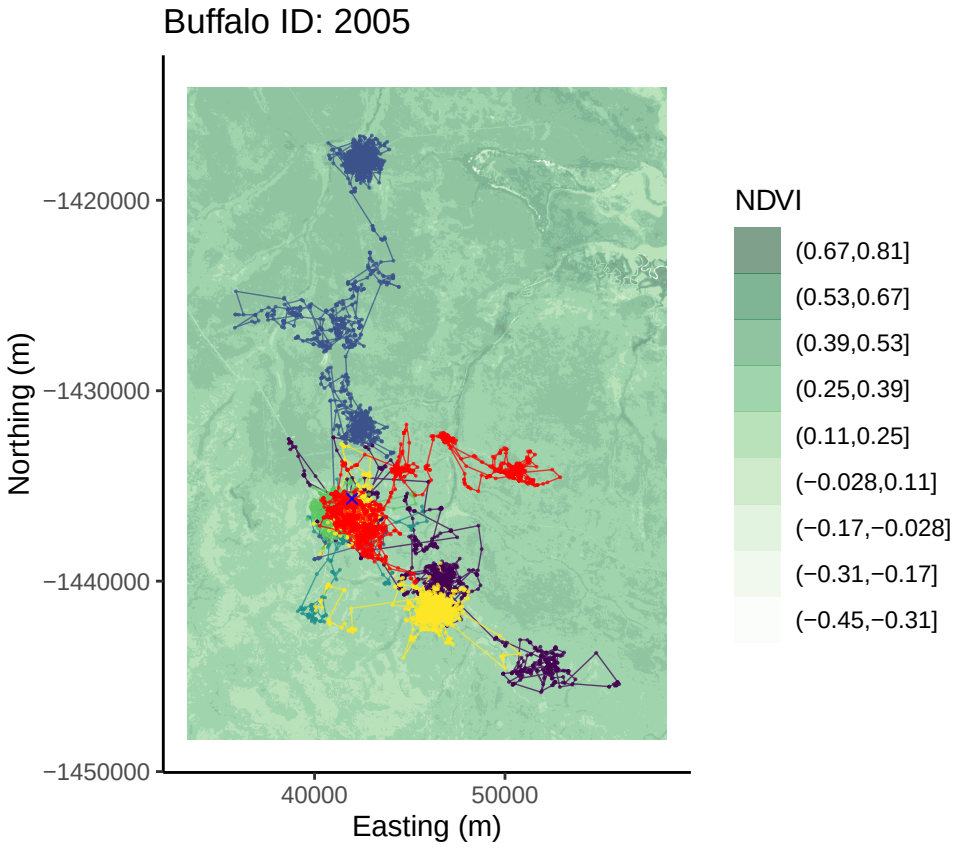
We are ordering the simulated data by the distance to the obsered starting location, and then taking the closest `n_sims` trajectories.

```
ndvi_dry_xy <- as.data.frame(ndvi_dry, xy = TRUE)
ndvi_dry_xy$ndvi_dry_discrete <- cut(ndvi_dry_xy$ndvi_dry, breaks=9, dig.lab = 2)

n_sims <- 5
buffer <- 2500

for(i in 1:length(buffalo_ids)) {
# for(i in 1:1) {

# find initial location for buffalo data
buffalo_id_initial_x <- buffalo_data %>%
  filter(id == buffalo_ids[i]) %>% slice(1) %>% pull(x_)
buffalo_id_initial_y <- buffalo_data %>%
  filter(id == buffalo_ids[i]) %>% slice(1) %>% pull(y_)
buffalo_id_initial_df <- data.frame("X" = buffalo_id_initial_x,
                                    "y" = buffalo_id_initial_y)

# find initial locations for simulated data
sim_traj <- sim_data_btime %>%
  filter(id == buffalo_ids[i]) %>%
  dplyr::group_by(traj) %>%
  slice(1) %>%
  pull(traj)

sim_traj_initial_x <- sim_data_btime %>% filter(id == buffalo_ids[i]) %>%
  dplyr::group_by(traj) %>% slice(1) %>% pull(x_)
sim_traj_initial_y <- sim_data_btime %>% filter(id == buffalo_ids[i]) %>%
  dplyr::group_by(traj) %>% slice(1) %>% pull(y_)

# calculate distance between the starting locations of the
# buffalo and simulated data
diff_x <- buffalo_id_initial_x - sim_traj_initial_x
diff_y <- buffalo_id_initial_y - sim_traj_initial_y
diff_xy <- sqrt(diff_x^2 + diff_y^2)

diff_xy_df <- data.frame(sim_traj,
                         sim_traj_initial_x,
                         sim_traj_initial_y,
                         diff_x, diff_y, diff_xy) %>% arrange(diff_xy)

# select the closest n_sims trajectories
sim_traj_xy_ids <- diff_xy_df %>% slice_head(n = n_sims) %>% pull(sim_traj)

# set the extent of the plot
extent_sim <- sim_data_btime %>%
```

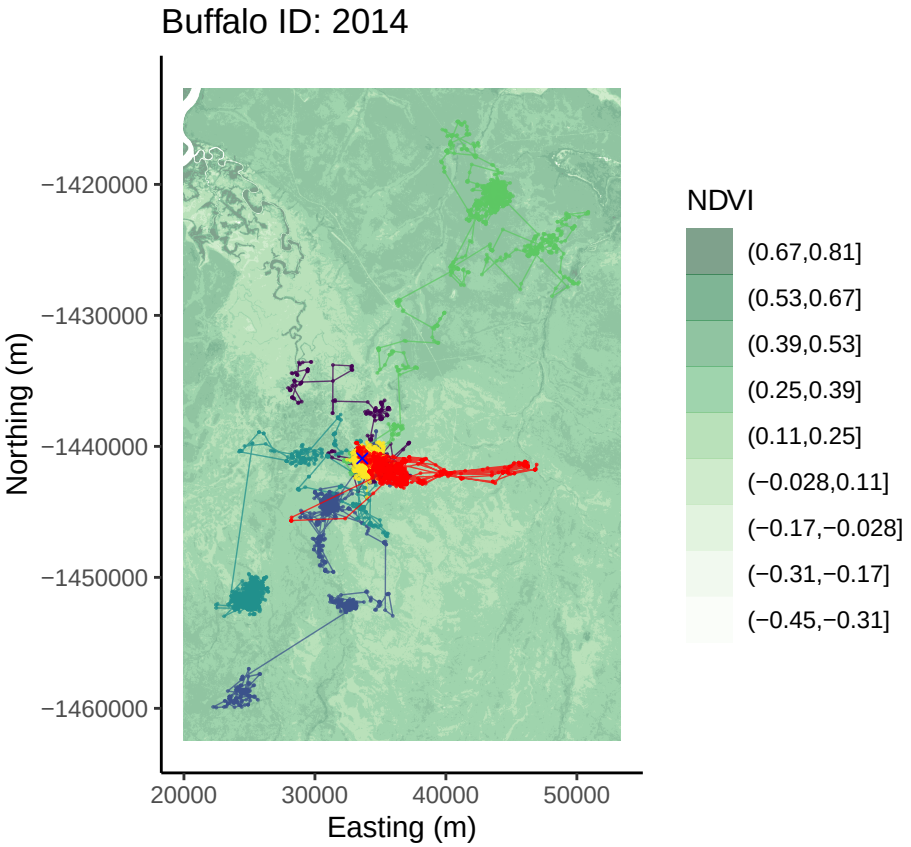
```
  filter(id == buffalo_ids[i] & traj %in% sim_traj_xy_ids) %>%
  summarise(min_x = min(x_), min_y = min(y_), max_x = max(x_), max_y = max(y_))

extent_buffalo <- buffalo_data %>% filter(id == buffalo_ids[i]) %>%
  summarise(min_x = min(x_), min_y = min(y_), max_x = max(x_), max_y = max(y_))


  print(ggplot() +
          geom_raster(data = ndvi_dry_xy,
                      aes(x = x, y = y, fill = ndvi_dry_discrete),
                      alpha = 0.5) +
          scale_fill_brewer("NDVI", palette = "Greens",
                            guide = guide_legend(reverse = TRUE)) +
          geom_path(data = sim_data_btime %>%
                      filter(id == buffalo_ids[i] & traj %in% sim_traj_xy_ids),
                    aes(x = x2_, y = y2_, colour = traj),
                    alpha = 0.75,
                    linewidth = 0.25) +
          geom_point(data = sim_data_btime %>%
                       filter(id == buffalo_ids[i] & traj %in% sim_traj_xy_ids),
                     aes(x = x2_, y = y2_, colour = traj),
                     alpha = 0.75,
                     size = 0.01) +
          geom_path(data = buffalo_data %>% filter(id == buffalo_ids[i]),
                    aes(x = x2_, y = y2_),
                    colour = "red",
                    alpha = 0.75,
                    linewidth = 0.25) +
          geom_point(data = buffalo_data %>% filter(id == buffalo_ids[i]),
                     aes(x = x2_, y = y2_),
                     colour = "red",
                     alpha = 0.75,
                     size = 0.01) +
          geom_point(data = buffalo_id_initial_df, aes(x = X, y = y),
                     colour = "blue",
                     alpha = 1,
                     shape = 4) +
          scale_color_viridis_d(guide = FALSE) +
          scale_x_continuous("Easting (m)",
                             limits = c(min(extent_sim[[1]], extent_buffalo[[1]])-buffer,
                                        max(extent_sim[[3]], extent_buffalo[[3]])+buffer)) +
          scale_y_continuous("Northing (m)",
                             limits = c(min(extent_sim[[2]], extent_buffalo[[2]])-buffer,
                                        max(extent_sim[[4]], extent_buffalo[[4]])+buffer)) +
          ggtitle(paste("Buffalo ID:", buffalo_ids[i])) +
          coord_equal() +
          theme_classic() +
          theme(legend.position = "right"))

  # ggsave(paste0("outputs/plots/sim_validation_memALL/dynamic_sim_trajectories_3p_id_",
  #               buffalo_ids[i], "_n_sims_", n_sims, "_", Sys.Date(), ".png"),
  #        width=150, height=90, units="mm", dpi = 300)
```

```
}
```

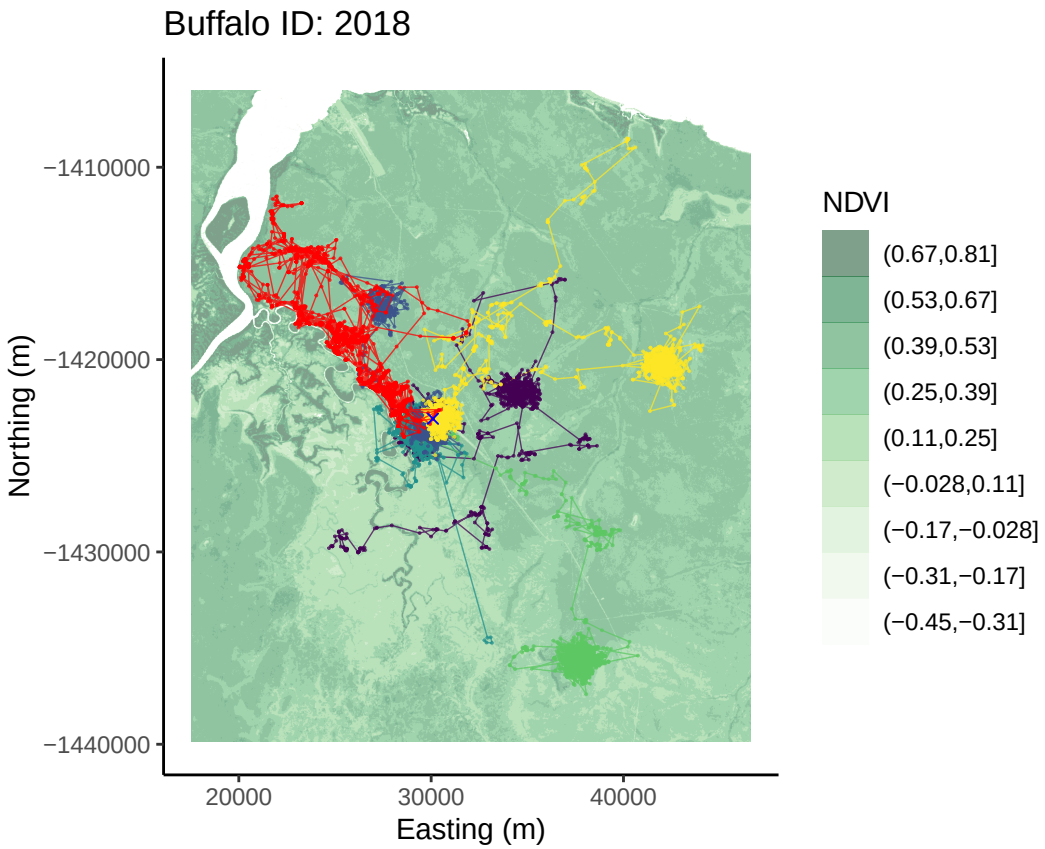
```
## Warning: Removed 3947444 rows containing missing values (`geom_raster()`).
```



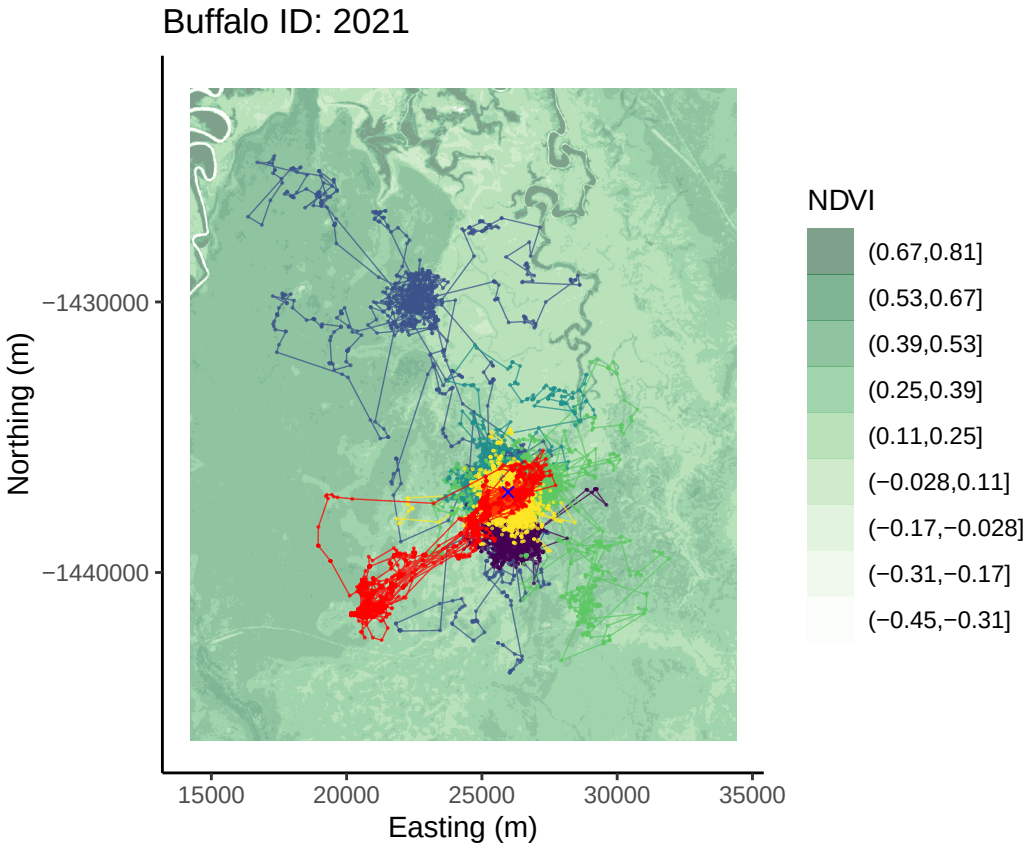

```
## Warning: Removed 2671771 rows containing missing values (`geom_raster()`).
```



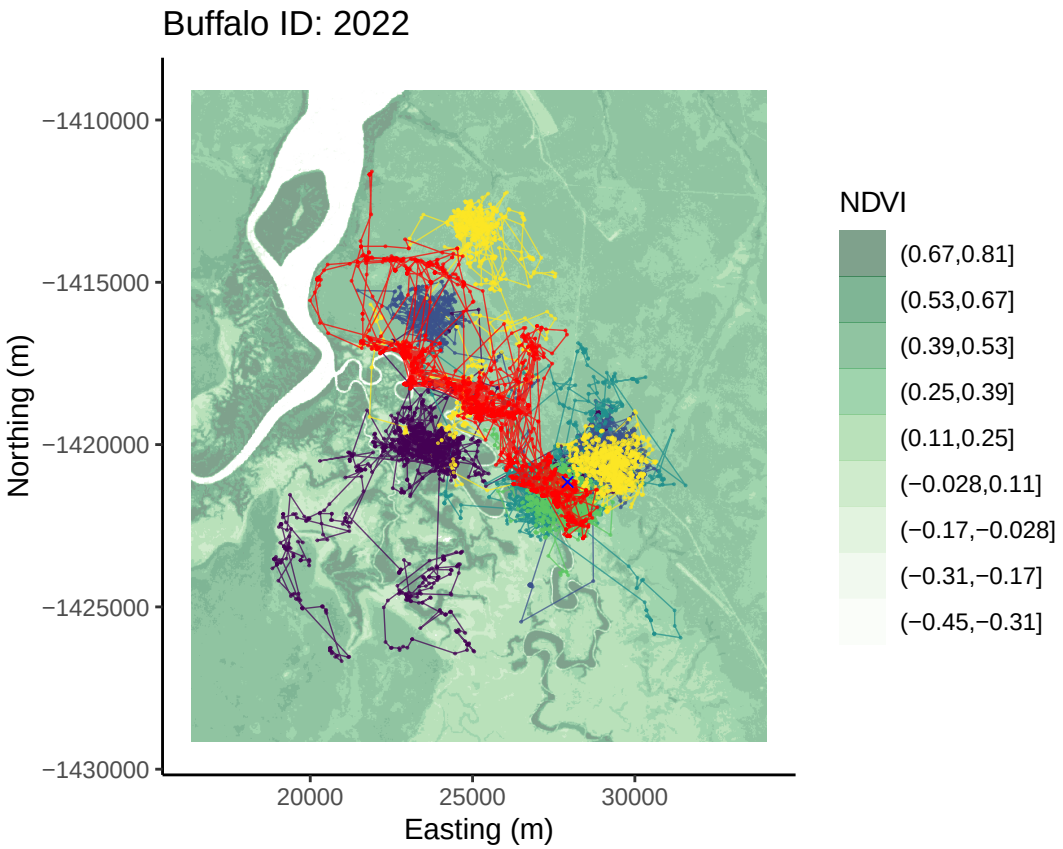

```
## Warning: Removed 3815503 rows containing missing values (`geom_raster()`).
```



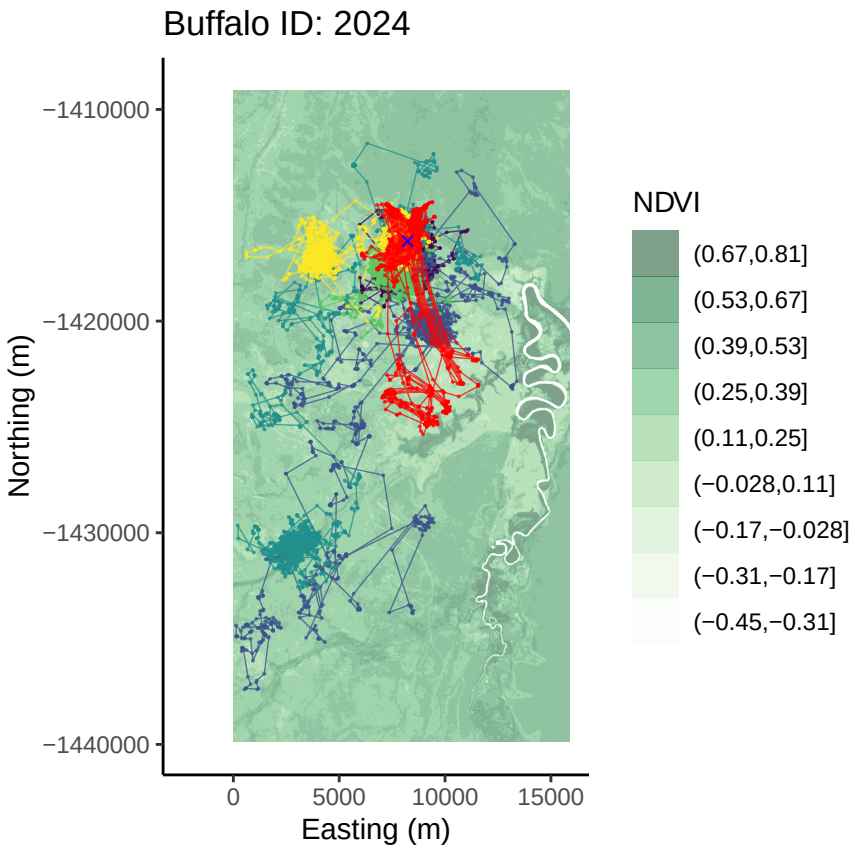

```
## Warning: Removed 4550114 rows containing missing values (‘geom_raster()‘).
```



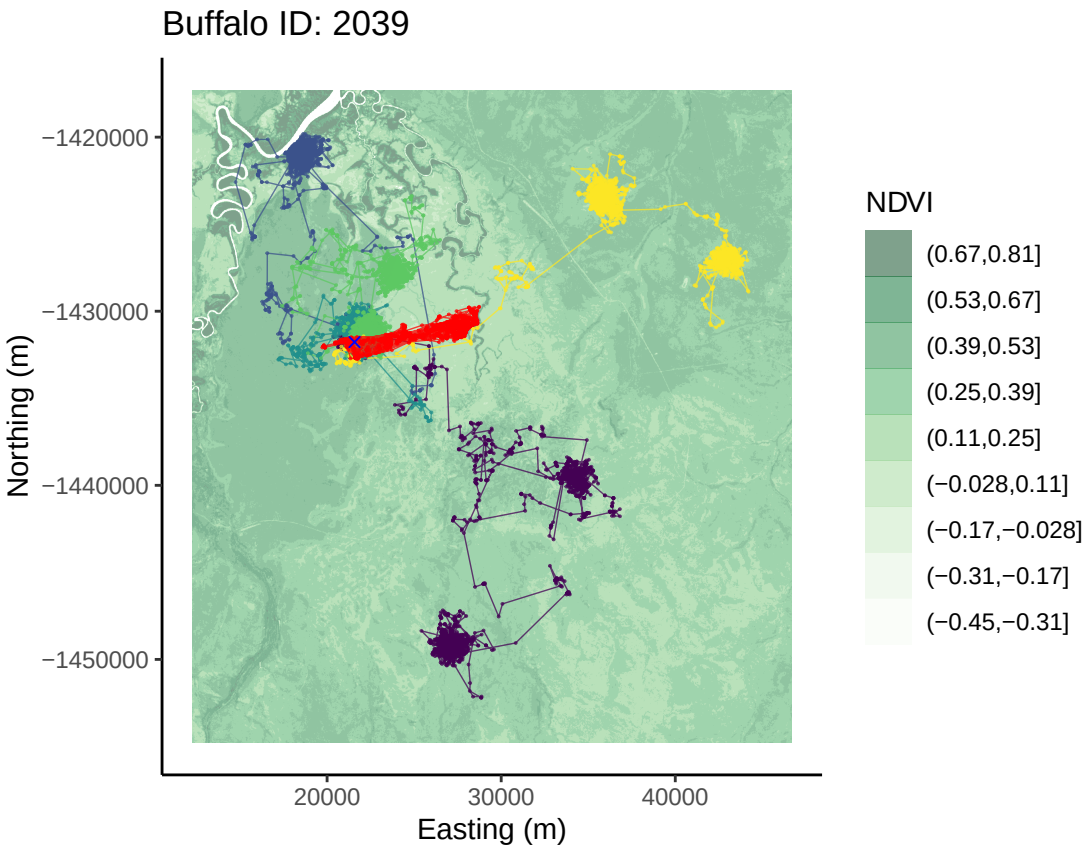

```
## Warning: Removed 4780421 rows containing missing values (‘geom_raster()‘).
```



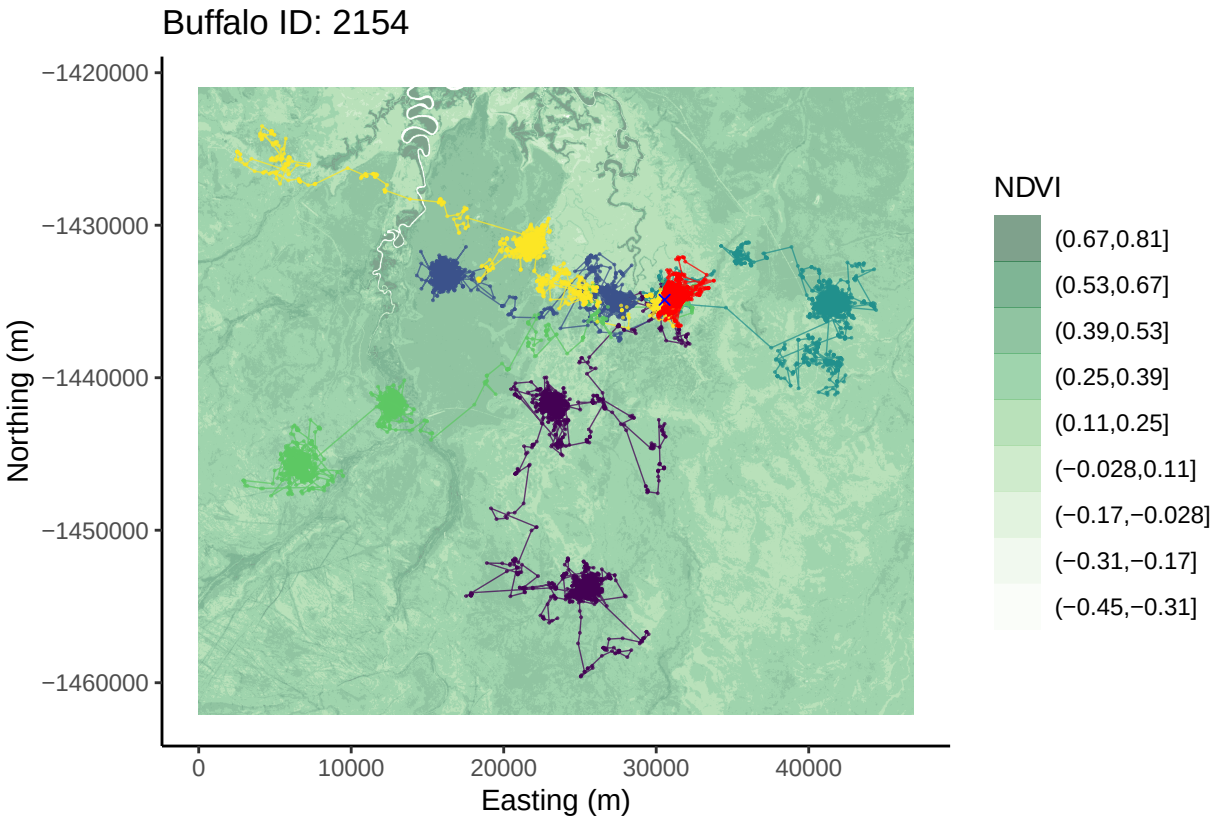

```
## Warning: Removed 4548382 rows containing missing values (‘geom_raster()‘).
```



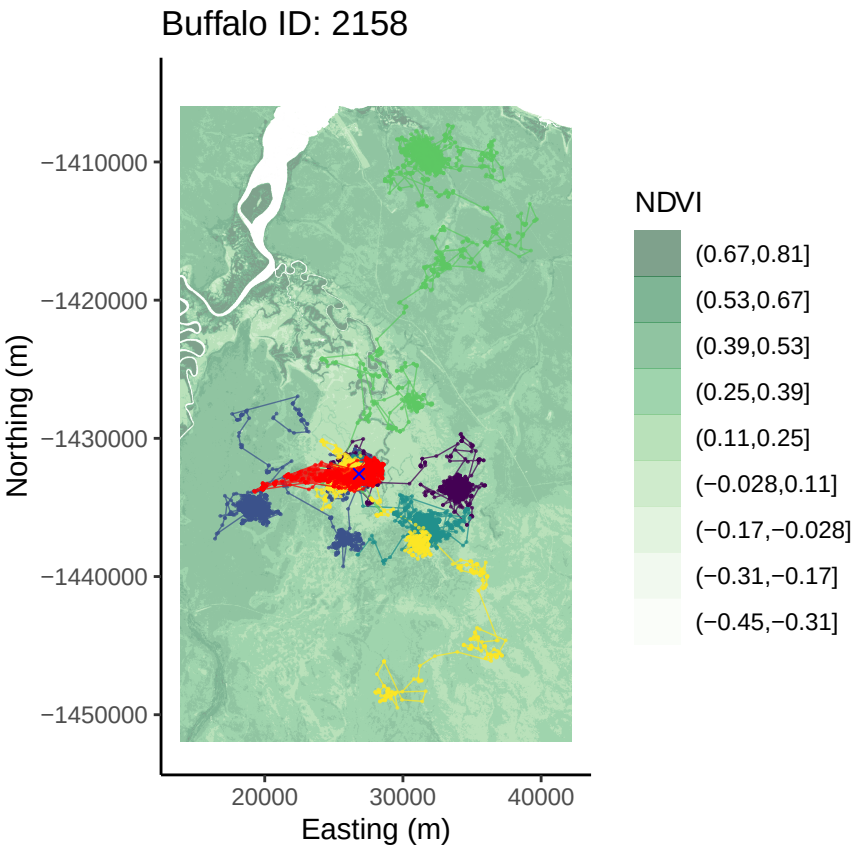

```
## Warning: Removed 3271060 rows containing missing values (‘geom_raster()‘).
```



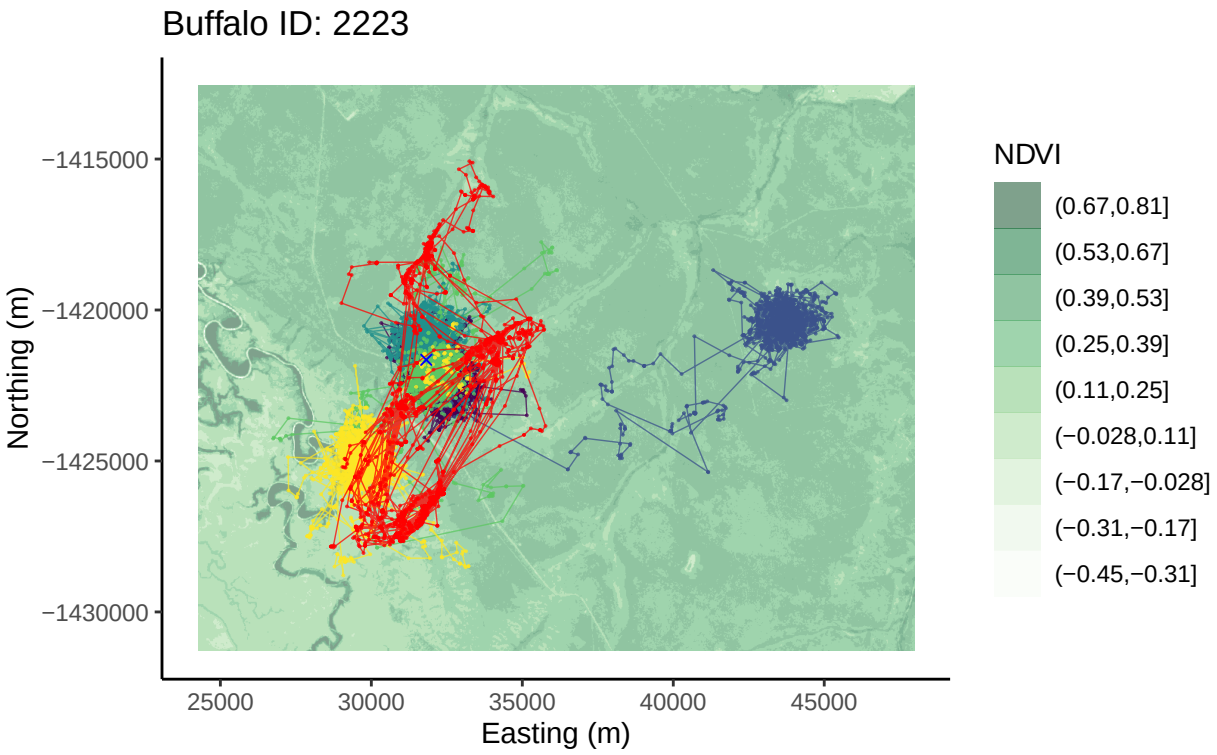

```
## Warning: Removed 2246731 rows containing missing values (‘geom_raster()‘).
```



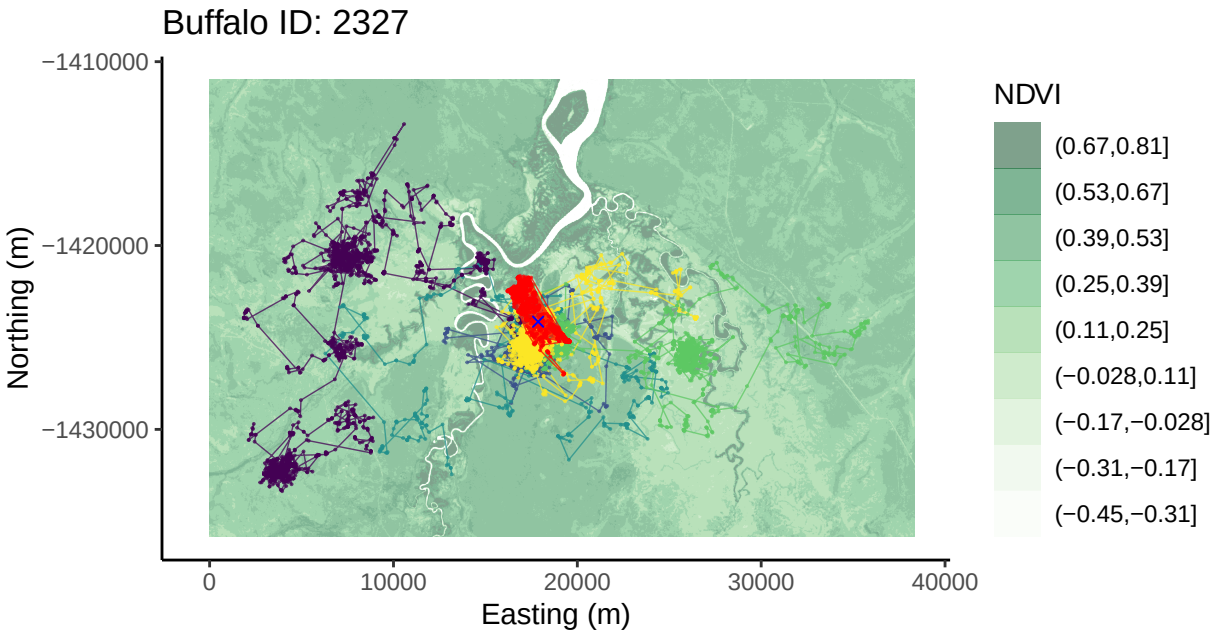

```
## Warning: Removed 3296460 rows containing missing values (‘geom_raster()‘).
```



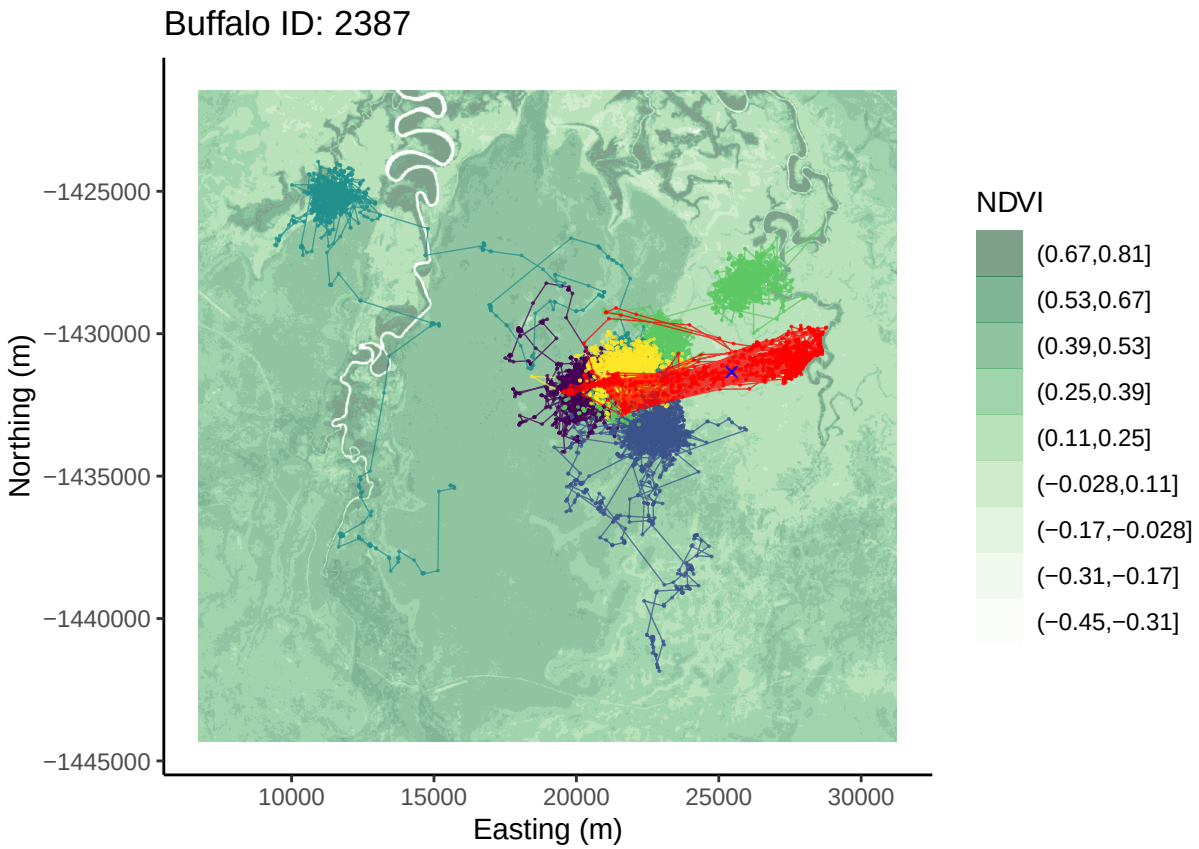

```
## Warning: Removed 4616839 rows containing missing values (‘geom_raster()‘).
```



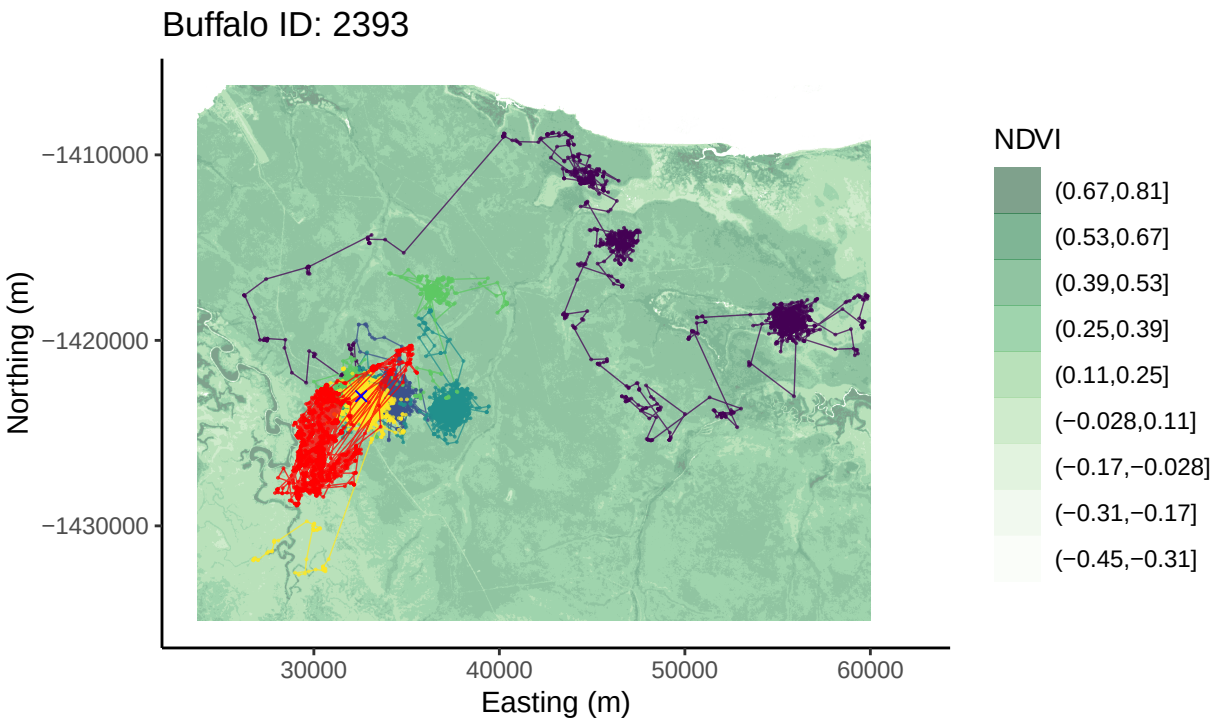

```
## Warning: Removed 3820373 rows containing missing values (‘geom_raster()‘).
```




```
## Warning: Removed 4432958 rows containing missing values (‘geom_raster()‘).
```




```
## Warning: Removed 3745779 rows containing missing values (`geom_raster()`).
```



Hourly movement behaviour and selection of covariates

Here we bin the trajectories into the hours of the day, and calculate the mean, median (where appropriate) and sd values for the step lengths and four habitat covariates.

We also save the results as a csv to compare between all of the models.

```
buffalo_hourly_habitat <-
  buffalo_data %>% dplyr::group_by(hour, id) %>%
  summarise(n = n(),
            step_length_mean = mean(sl_),
            step_length_median = median(sl_),
            step_length_sd = sd(sl_),
            ndvi_mean = mean(ndvi_dry),
            ndvi_median = median(ndvi_dry),
            ndvi_sd = sd(ndvi_dry),
            herby_mean = mean(veg_herby),
            herby_sd = sd(veg_herby),
            canopy_mean = mean(canopy_cover/100),
            canopy_sd = sd(canopy_cover/100),
            slope_mean = mean(slope),
            slope_median = median(slope),
            slope_sd = sd(slope)
            ) %>% ungroup()
```

```
## `summarise()` has grouped output by 'hour'. You can override using the `.groups` argument.
```

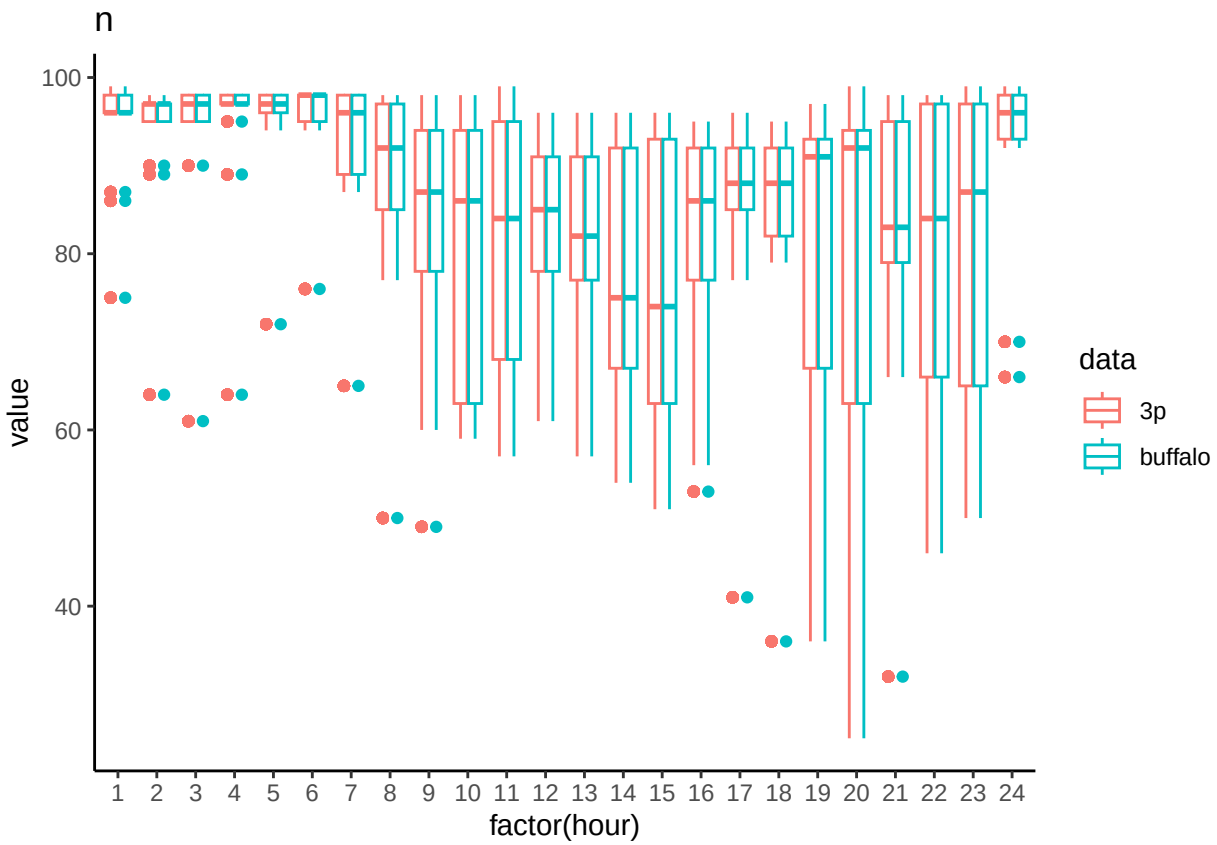
```r
buffalo_hourly_habitat <- data.frame("data" = "buffalo", buffalo_hourly_habitat) %>%
  mutate(id = as.factor(id))

write.csv(buffalo_hourly_habitat,
          paste0("outputs/buffalo_summaries_hourly_habitat_", Sys.Date(), ".csv"))

buffalo_hourly_habitat_long <- buffalo_hourly_habitat %>%
  pivot_longer(cols = !c(data, hour, id), names_to = "variable", values_to = "value")

sim_hourly_habitat <-
  sim_data_btime %>% dplyr::group_by(hour, traj) %>%
  summarise(n = n(),
            step_length_mean = mean(sl_),
            step_length_median = median(sl_),
            step_length_sd = sd(sl_),
            ndvi_mean = mean(ndvi_dry),
            ndvi_median = median(ndvi_dry),
            ndvi_sd = sd(ndvi_dry),
            herby_mean = mean(veg_herby),
            herby_sd = sd(veg_herby),
            canopy_mean = mean(canopy_cover/100),
            canopy_sd = sd(canopy_cover/100),
            slope_mean = mean(slope_raster),
            slope_median = median(slope_raster),
            slope_sd = sd(slope_raster)
            ) %>% ungroup()
```

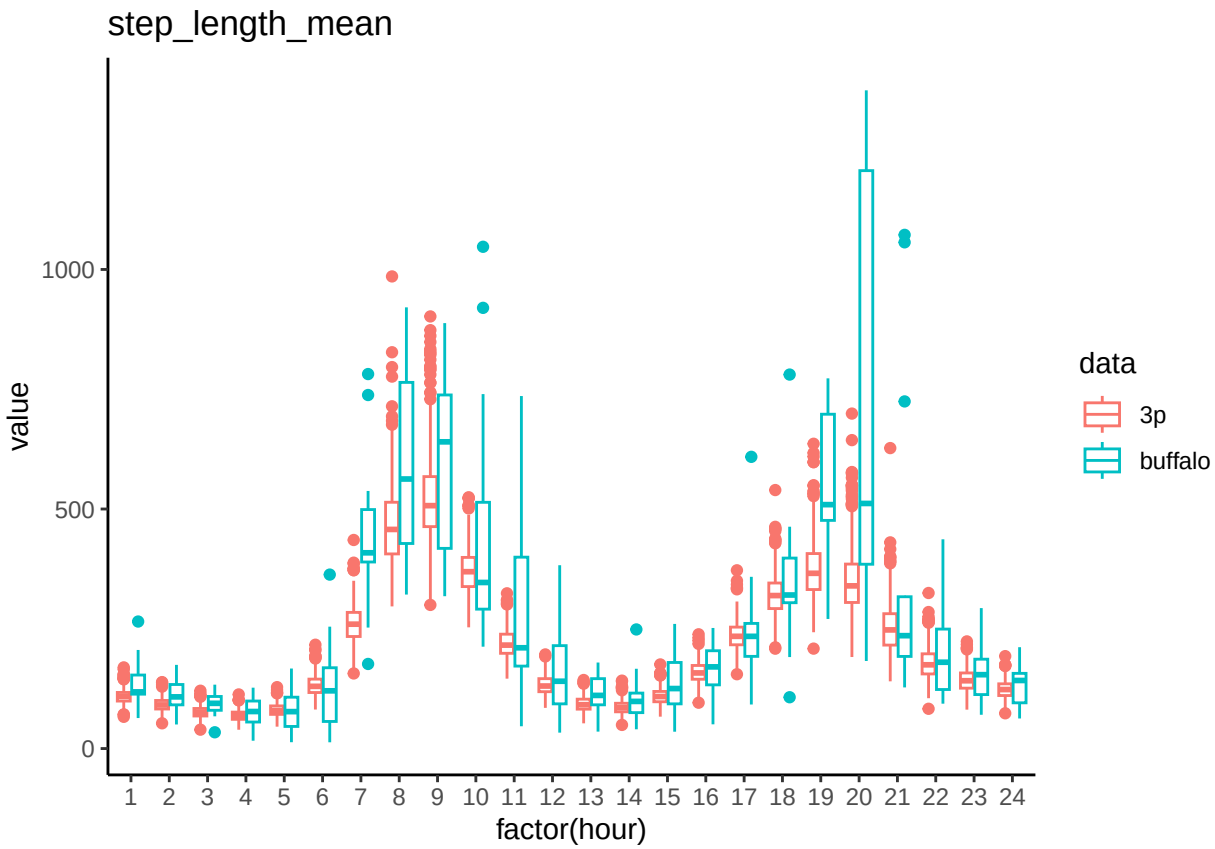
```
## `summarise()` has grouped output by 'hour'. You can override using the `.groups` argument.
```

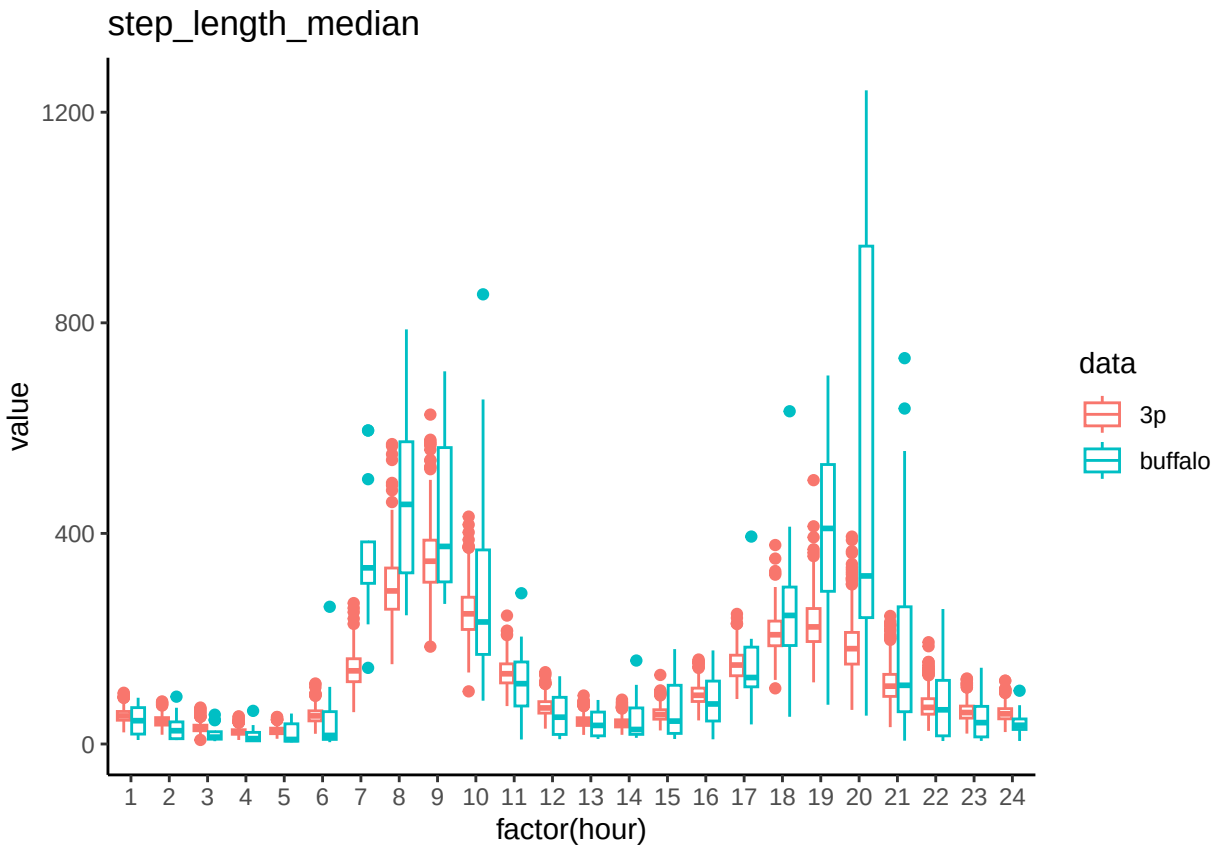
```r
sim_hourly_habitat <- data.frame("data" = "3p", sim_hourly_habitat) %>%
  rename(id = traj) %>% mutate(id = as.factor(id))

write.csv(sim_hourly_habitat,
          paste0("outputs/sim_3p_memALL_summaries_hourly_habitat_", Sys.Date(), ".csv"))

sim_hourly_habitat_long <- sim_hourly_habitat %>%
  pivot_longer(cols = !c(data, hour, id), names_to = "variable", values_to = "value")

# combine the dataframe
hourly_habitat_long <- bind_rows(buffalo_hourly_habitat_long, sim_hourly_habitat_long)
```

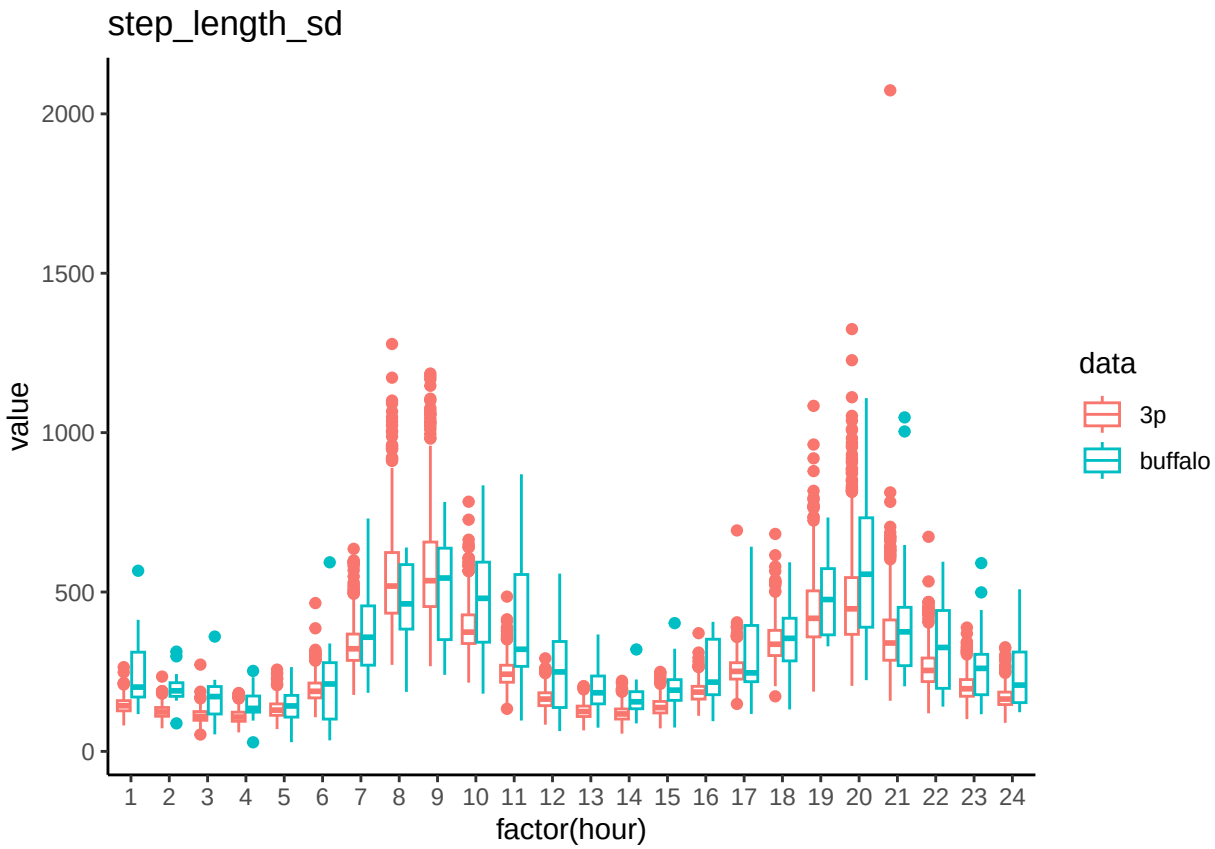
Plotting the hourly habitat selection

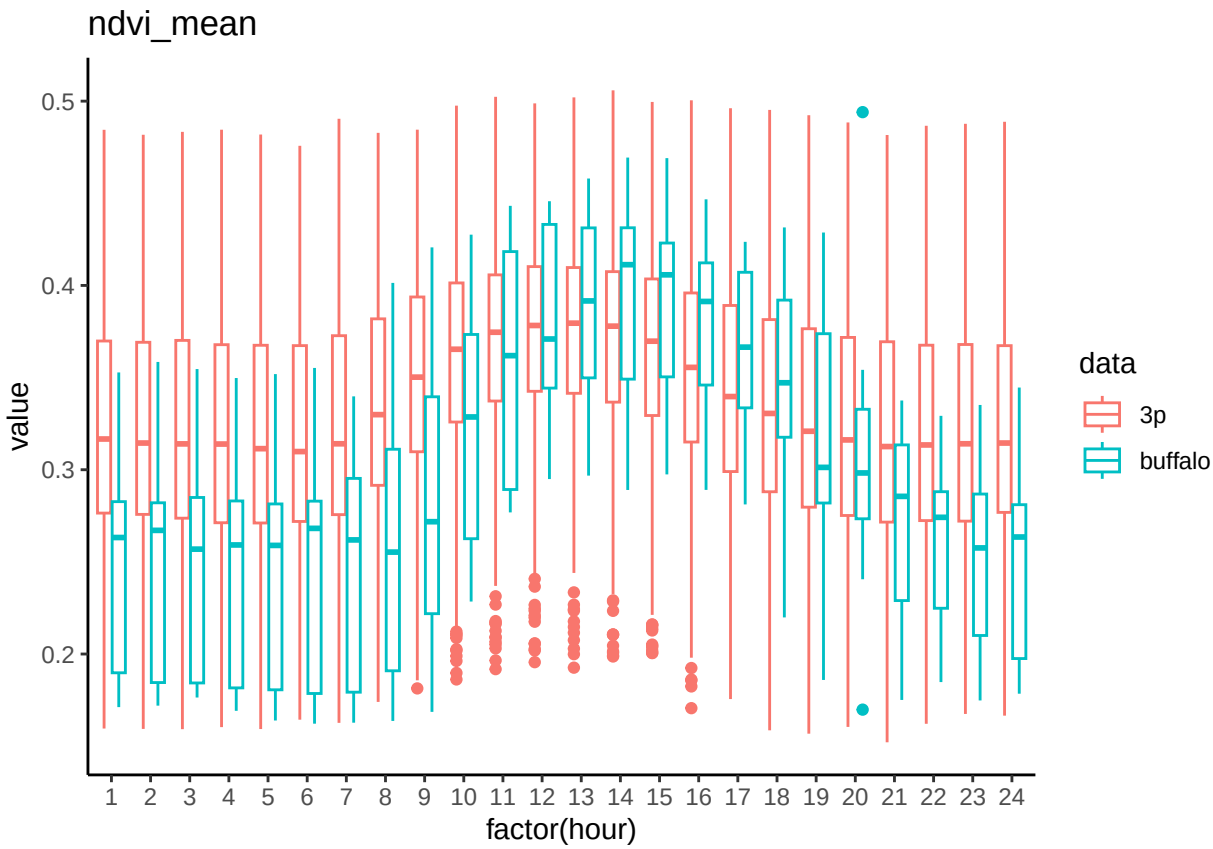
```r
for(i in 1:length(unique(hourly_habitat_long$variable))) {

  print(ggplot(data = hourly_habitat_long %>%
                 filter(variable == unique(variable)[i]),
               aes(x = factor(hour), y = value, colour = data)) +
          geom_boxplot() +
          ggtitle(unique(hourly_habitat_long$variable)[i]) +
          theme_classic())

}
```

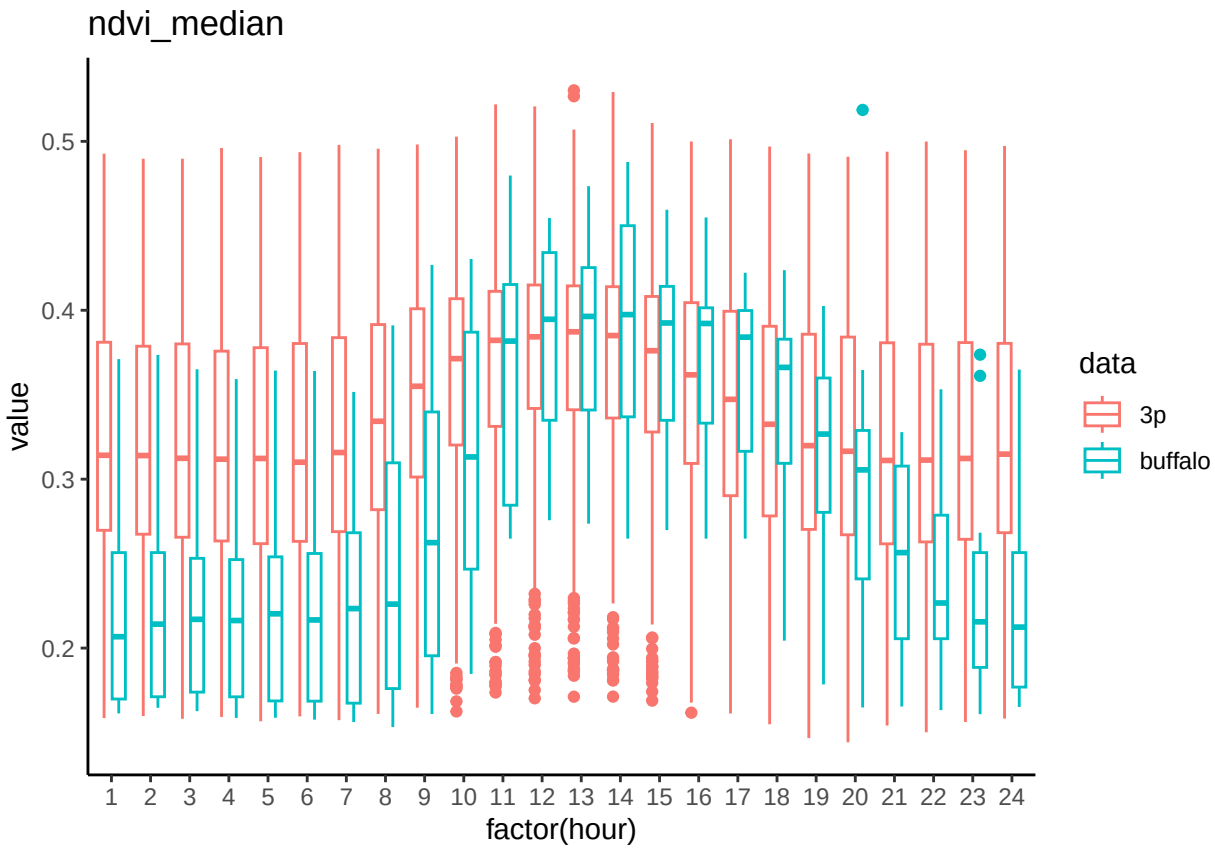

n
100
80
60
40
value
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

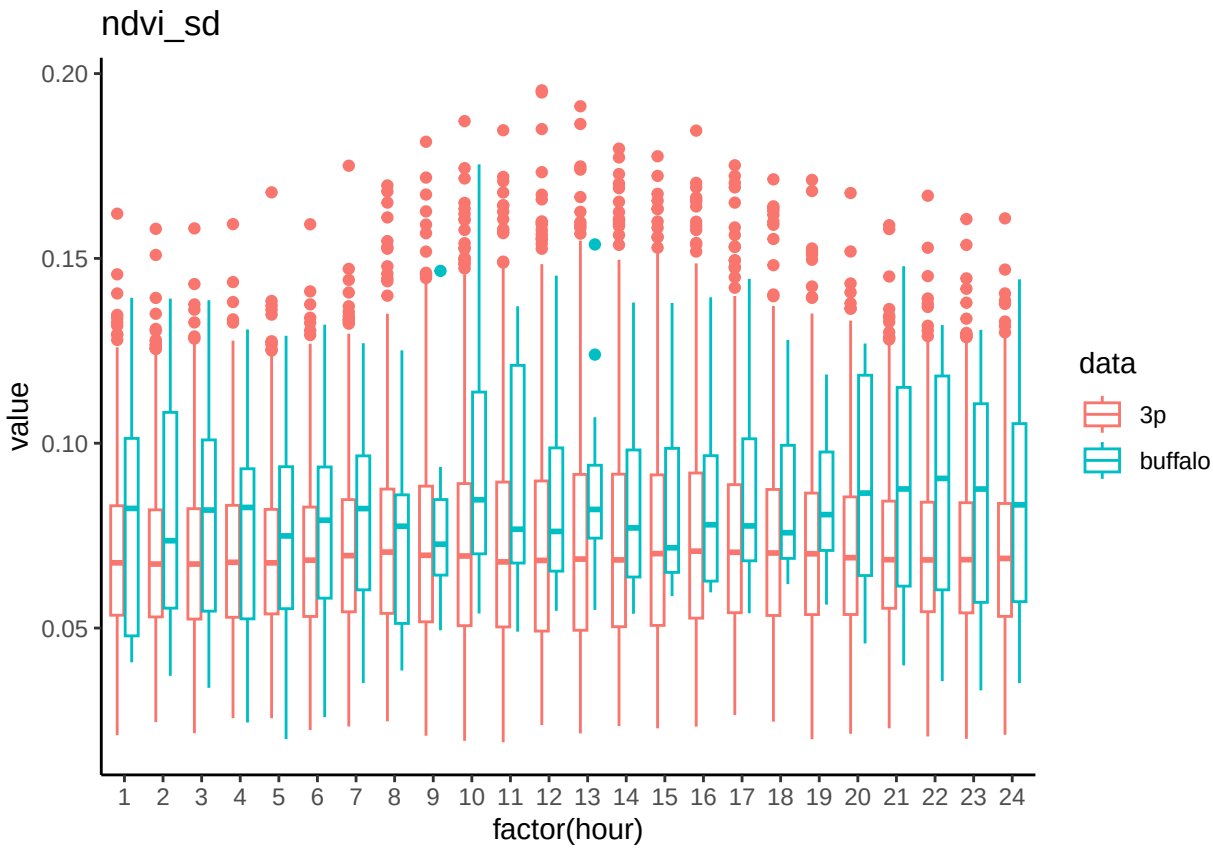

step_length_mean
value
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

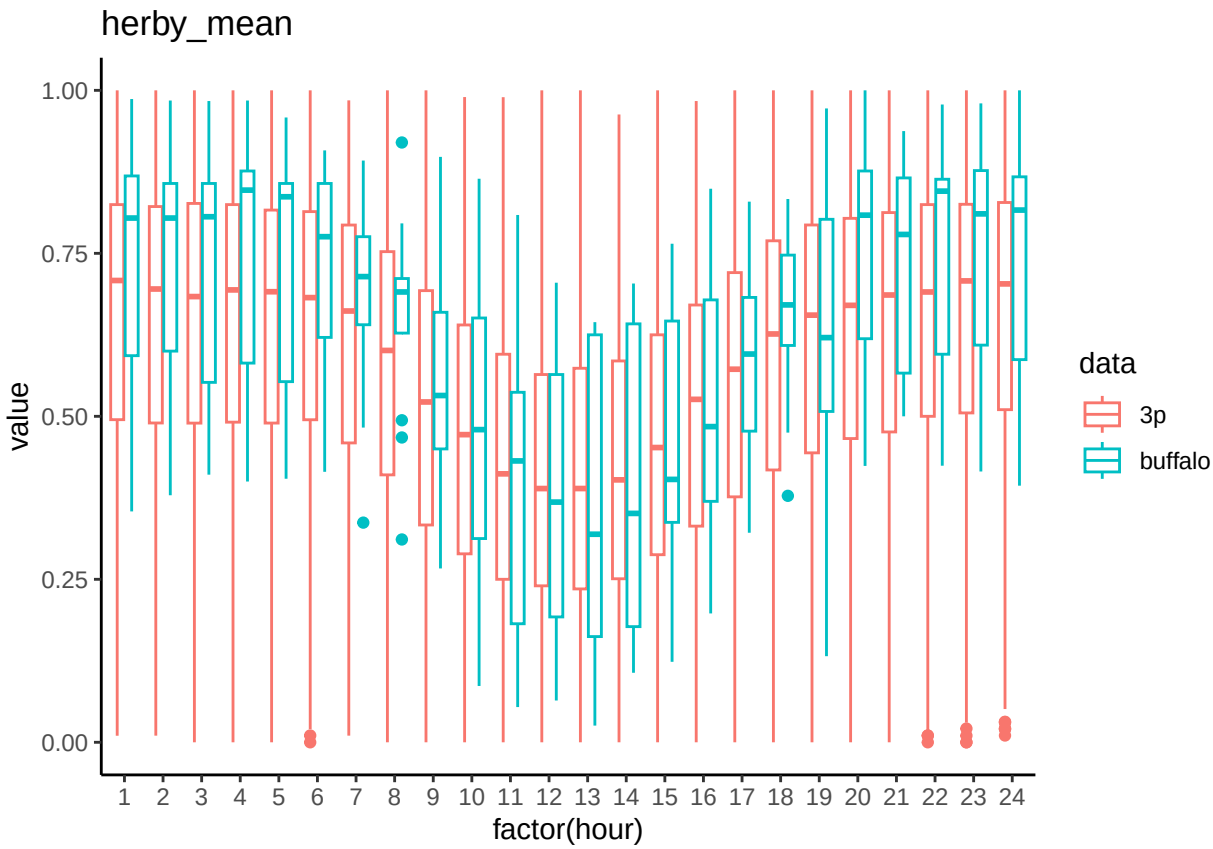

step_length_median
value
1200
800
400
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

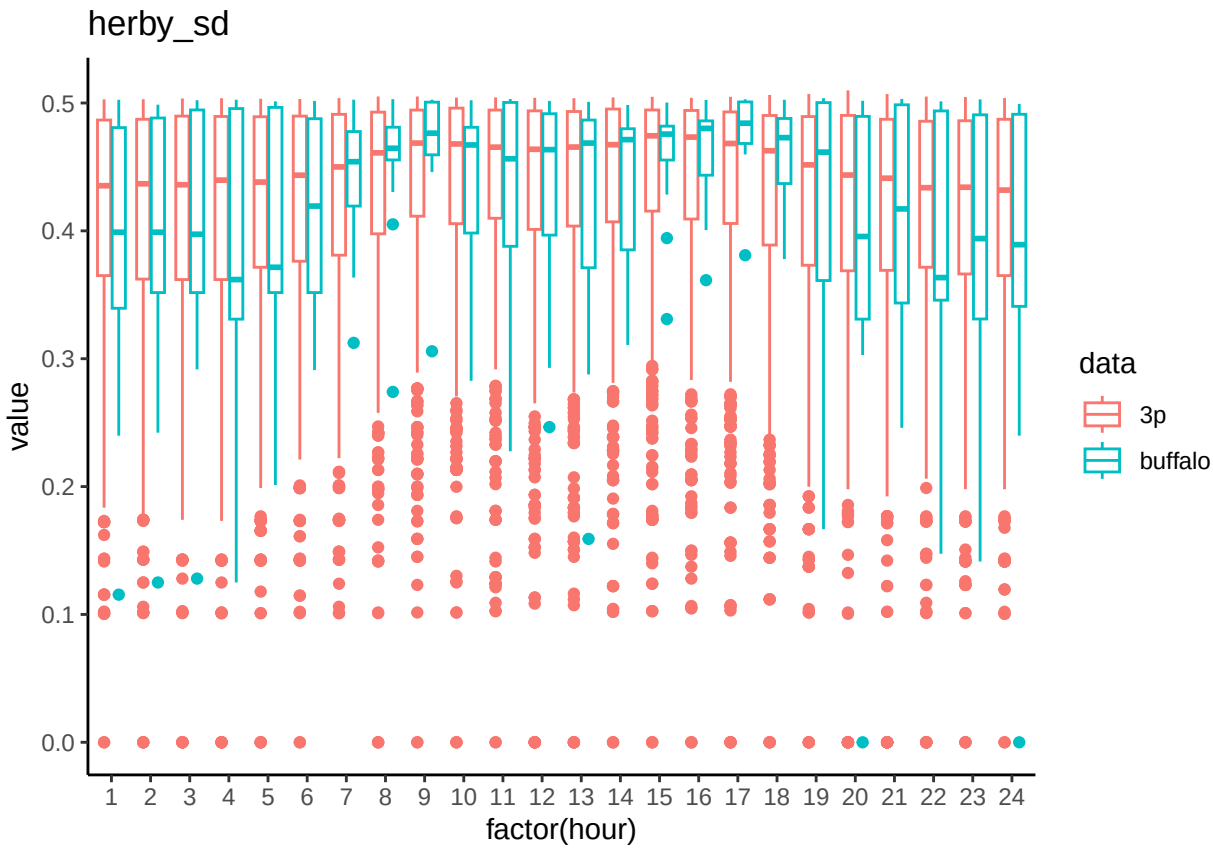

step_length_sd
value
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

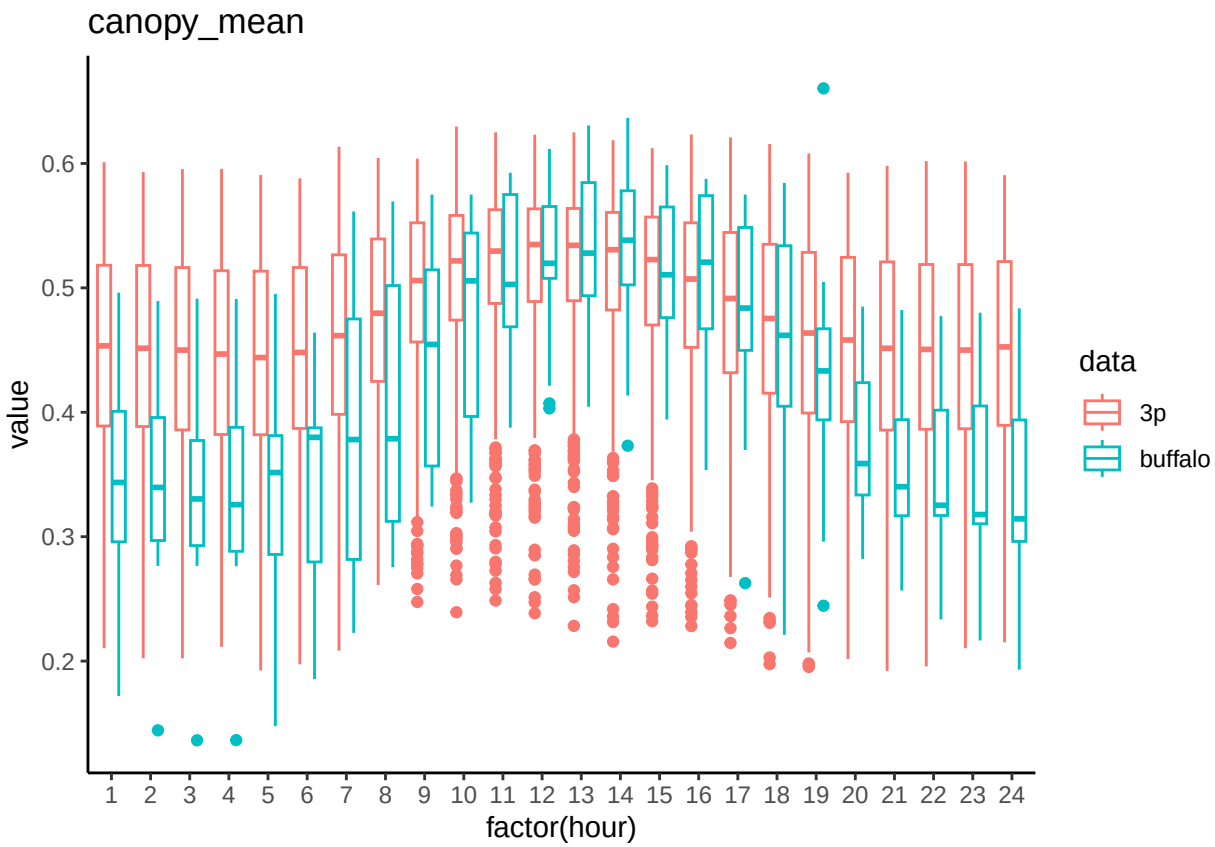

ndvi_mean
value
0.5
0.4
0.3
0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

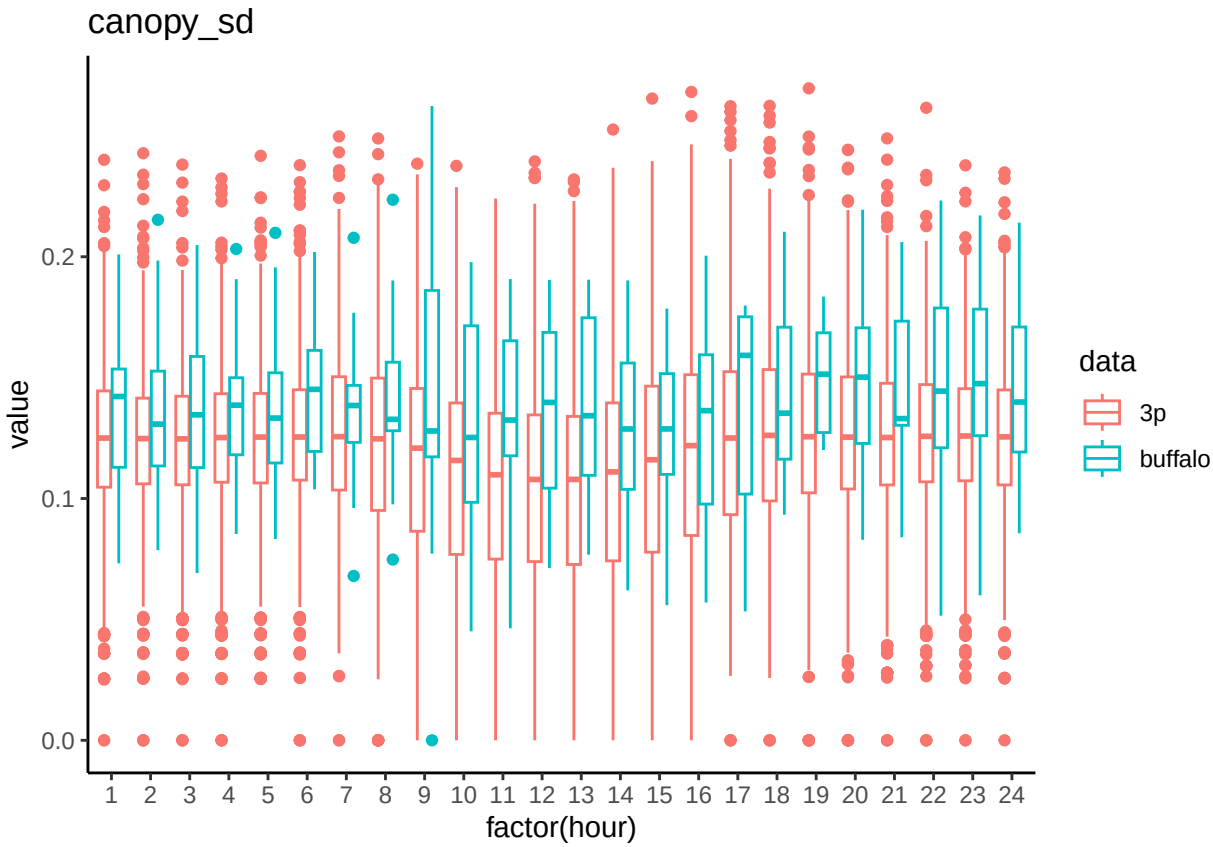

ndvi_median
value
0.5
0.4
0.3
0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

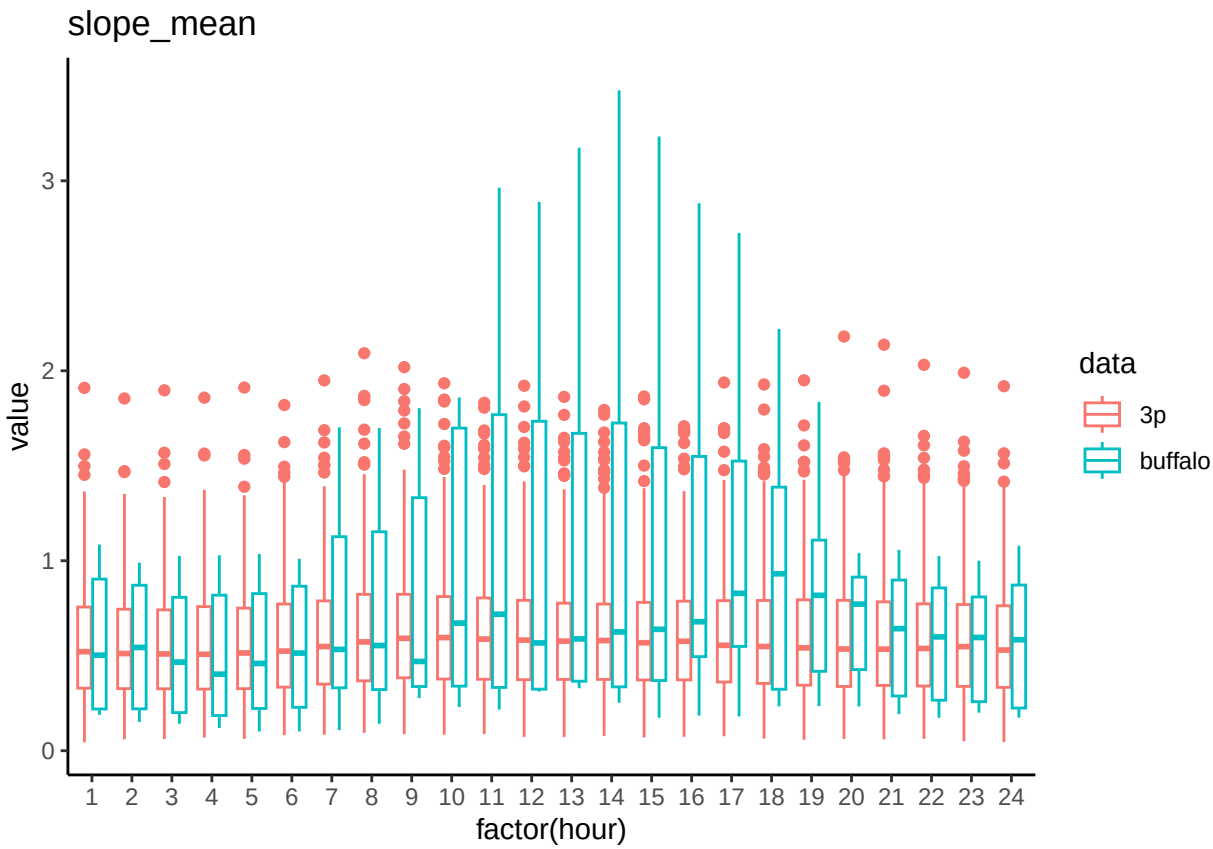

ndvi_sd
0.20
0.15
0.10
0.05
value
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

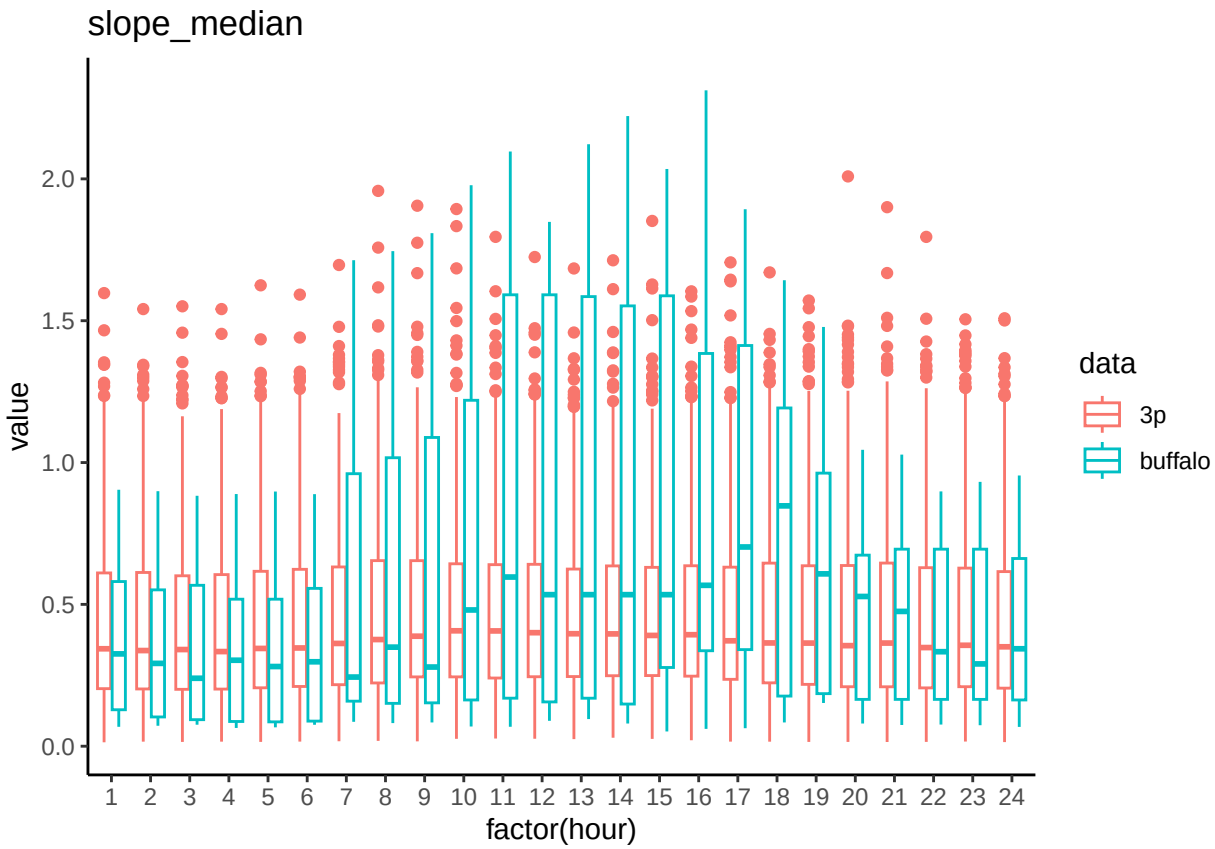

herby_mean
value
1.00
0.75
0.50
0.25
0.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo

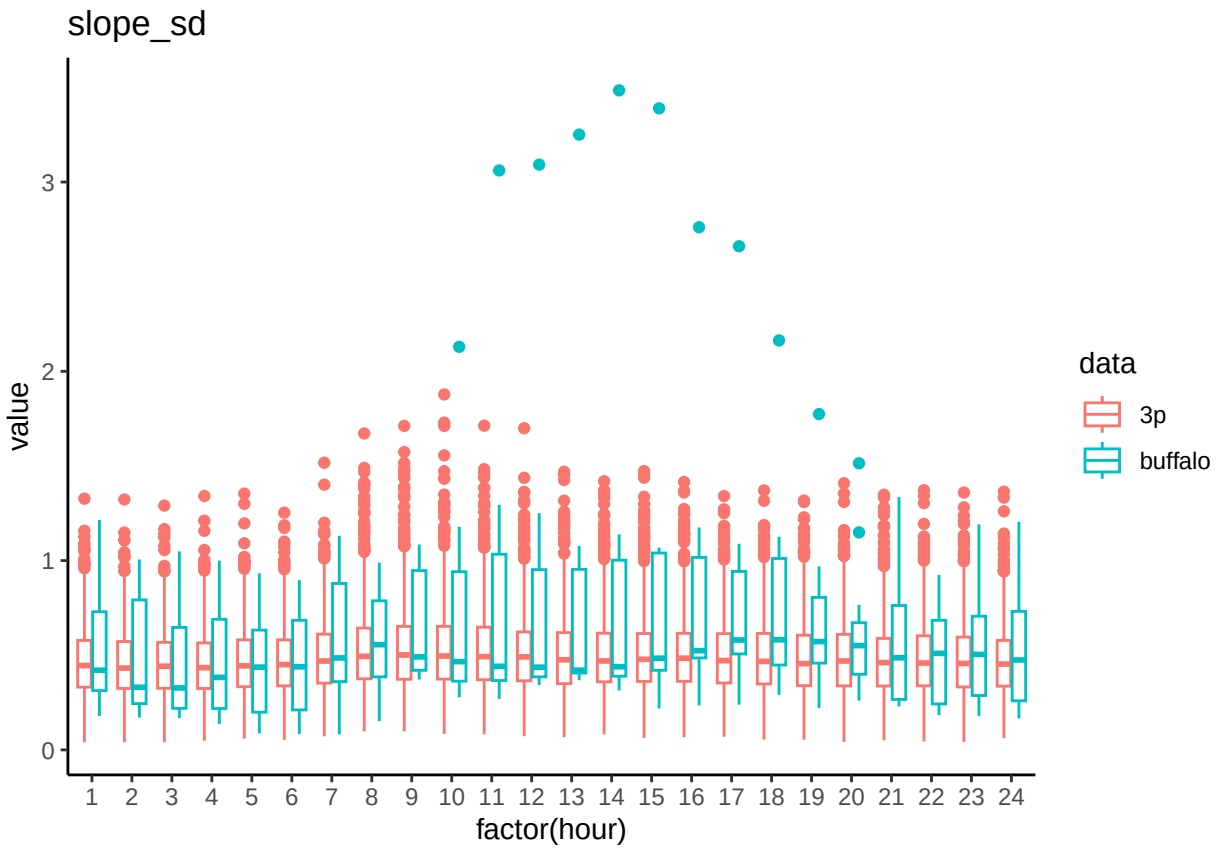

herby_sd
value
0.0
0.1
0.2
0.3
0.4
0.5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo


canopy_mean
value
0.6
0.5
0.4
0.3
0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo


canopy_sd
value
0.0
0.1
0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo


slope_mean
value
0
1
2
3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo


slope_median
value
2.0
1.5
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
factor(hour)
data
3p
buffalo



Summary statistics for the full trajectory

Convert buffalo data to ltraj object to calculate residence time

```
buffalo_mean_sl <- mean(buffalo_data$sl_)

buffalo_data_ltraj <- amt::as_ltraj(buffalo_data, id = buffalo_data$id)
buffalo_RT <- residenceTime(buffalo_data_ltraj,
                            radius = buffalo_mean_sl,
                            maxt = 12, units = "hours", addinfo = FALSE)
```

Calculate summary statistics for buffalo

Here we've created a loop that contains the summary statistics. We subset each individual animal's trajectory and then below each simulated individual's trajectory and calculate values for all of the summary statistics.

```
buffer <- 10000
res <- 100

# setup empty objects to store the results
id <- c()

step_length_median <- c()
step_length_mean <- c()
ndvi_mean <- c()
ndvi_median <- c()
```

```r
ndvi_sd <- c()
herby_mean <- c()
herby_sd <- c()
canopy_mean <- c()
canopy_sd <- c()
slope_mean <- c()
slope_median <- c()
slope_sd <- c()

gamma_shape <- c()
gamma_scale <- c()
vm_kappa <- c()

straightness <- c()
cum_dist <- c()
tot_dist <- c()
msd <- c()
intensity_use <- c()
sinuosity <- c()
tac <- c()

residence_time <- c() # residence time in hours

hr_area_50 <- c()
hr_area_75 <- c()
hr_area_95 <- c()
hr_monthly_overlap_ba <- c()
hr_monthly_overlap_vi <- c()


tic()

for(k in 1:length(buffalo_ids)) {

  buffalo_data_id <- buffalo_data %>% filter(id == buffalo_ids[k])

  xmin <- min(buffalo_data_id$x2_) - buffer
  xmax <- max(buffalo_data_id$x2_) + buffer
  ymin <- min(buffalo_data_id$y2_) - buffer
  ymax <- max(buffalo_data_id$y2_) + buffer
  template_raster <- rast(xmin = xmin, xmax = xmax,
                          ymin = ymin, ymax = ymax,
                          res = res, crs = crs("epsg:3112"))

  id[k] <- buffalo_data_id$id[1]

  step_length_median[k] <- median(buffalo_data_id$sl_)
  step_length_mean[k] <- mean(buffalo_data_id$sl_)
  ndvi_mean[k] <- mean(buffalo_data_id$ndvi_dry)
  ndvi_median[k] <- median(buffalo_data_id$ndvi_dry)
  ndvi_sd[k] <- sd(buffalo_data_id$ndvi_dry)
  herby_mean[k] <- mean(buffalo_data_id$veg_herby)
  herby_sd[k] <- sd(buffalo_data_id$veg_herby)
```

```r
canopy_mean[k] <- mean(buffalo_data_id$canopy_cover/100)
canopy_sd[k] <- sd(buffalo_data_id$canopy_cover/100)
slope_mean[k] <- mean(buffalo_data_id$slope)
slope_median[k] <- median(buffalo_data_id$slope)
slope_sd[k] <- sd(buffalo_data_id$slope)

gamma_fit <- fit_distr(buffalo_data_id$sl_, "gamma")
gamma_shape[k] <- gamma_fit$params$shape
gamma_scale[k] <- gamma_fit$params$scale

vM_fit <- fit_distr(buffalo_data_id$sl_, "vonmises")
vm_kappa[k] <- vM_fit$params$kappa

straightness[k] <- amt::straightness(buffalo_data_id)
cum_dist[k] <- amt::cum_dist(buffalo_data_id)
tot_dist[k] <- amt::tot_dist(buffalo_data_id)
msd[k] <- amt::msd(buffalo_data_id)
intensity_use[k] <- amt::intensity_use(buffalo_data_id)
sinuosity[k] <- amt::sinuosity(buffalo_data_id)
tac[k] <- amt::tac(buffalo_data_id)

residence_time[k] <- mean(buffalo_RT[[k]][,2], na.rm = TRUE)/60/60

buffalo_hr_kde <- hr_kde(buffalo_data_id, trast = template_raster,
                         levels = c(0.5, 0.75, 0.95))
buffalo_hr_kde_area <- hr_area(buffalo_hr_kde)

hr_area_50[k] = buffalo_hr_kde_area[which(buffalo_hr_kde_area$level == 0.5),]$area/1e6
hr_area_75[k] = buffalo_hr_kde_area[which(buffalo_hr_kde_area$level == 0.75),]$area/1e6
hr_area_95[k] = buffalo_hr_kde_area[which(buffalo_hr_kde_area$level == 0.95),]$area/1e6


# there are three full months of data, so we can calculate the
# overlap between monthly home ranges for these three months

# kde 95% home ranges for each month
buffalo_hr_month8 <- hr_kde(buffalo_data_id %>% filter(month == 8),
                            trast = template_raster, levels = 0.95)
buffalo_hr_month9 <- hr_kde(buffalo_data_id %>% filter(month == 9),
                            trast = template_raster, levels = 0.95)
buffalo_hr_month10 <- hr_kde(buffalo_data_id %>% filter(month == 10),
                             trast = template_raster, levels = 0.95)

# calculate the overlap between monthly KDE home ranges and take the average
buffalo_hr_overlap_ba_8_9 <- hr_overlap(buffalo_hr_month8,
                                        buffalo_hr_month9, type = "ba")
buffalo_hr_overlap_ba_9_10 <- hr_overlap(buffalo_hr_month9,
                                         buffalo_hr_month10, type = "ba")
hr_monthly_overlap_ba[k] <- mean(c(buffalo_hr_overlap_ba_8_9$overlap,
                                   buffalo_hr_overlap_ba_9_10$overlap))

buffalo_hr_overlap_vi_8_9 <- hr_overlap(buffalo_hr_month8,
                                        buffalo_hr_month9, type = "vi")
```

```
  buffalo_hr_overlap_vi_9_10 <- hr_overlap(buffalo_hr_month9,
                                           buffalo_hr_month10, type = "vi")
  hr_monthly_overlap_vi[k] <- mean(c(buffalo_hr_overlap_vi_8_9$overlap,
                                     buffalo_hr_overlap_vi_9_10$overlap))

}

toc()
```

```
## 21.71 sec elapsed
```

Create a data frame combining all summaries

```
# create a data frame that has traj, id, and all the summaries
 buffalo_summary_df <- data.frame(

  traj = "obs",
  id = as.numeric(id),
  data = "obs",
  sim = "obs",

  step_length_median =  step_length_median,
  step_length_mean =  step_length_mean,
  ndvi_mean =  ndvi_mean,
  ndvi_median =  ndvi_median,
  ndvi_sd =  ndvi_sd,
  herby_mean =  herby_mean,
  herby_sd =  herby_sd,
  canopy_mean =  canopy_mean,
  canopy_sd =  canopy_sd,
  slope_mean =  slope_mean,
  slope_median =  slope_median,
  slope_sd =  slope_sd,

  gamma_shape =  gamma_shape,
  gamma_scale =  gamma_scale,
  vm_kappa =  vm_kappa,

  straightness =  straightness,
  cum_dist =  cum_dist,
  tot_dist =  tot_dist,
  msd =  msd,
  intensity_use =  intensity_use,
  sinuosity =  sinuosity,
  tac =  tac,
  residence_time =  residence_time,

  hr_area_50 = hr_area_50,
  hr_area_75 = hr_area_75,
  hr_area_95 = hr_area_95,
  hr_monthly_overlap_ba = hr_monthly_overlap_ba,
  hr_monthly_overlap_vi = hr_monthly_overlap_vi)
```

Write the buffalo summary statistics to a csv file

```
write_csv(buffalo_summary_df,
          paste0("outputs/buffalo_summary_statistics_df_", Sys.Date(), ".csv"))
```

Calculate summary statistics for simulated data

Convert simulated data to ltraj object to calculate residence time

```
sim_mean_sl <- mean(sim_data_btime$sl_)

sim_data_ltraj <- amt::as_ltraj(sim_data_btime, id = sim_data_btime$traj)
sim_RT <- residenceTime(sim_data_ltraj, radius = sim_mean_sl, maxt = 12,
                        units = "hours", addinfo = FALSE)
```

Setting up a loop for each simulated trajectory

```
buffer <- 10000
res <- 100

# setup empty objects to store the results
id <- c()

step_length_median <- c()
step_length_mean <- c()
ndvi_mean <- c()
ndvi_median <- c()
ndvi_sd <- c()
herby_mean <- c()
herby_sd <- c()
canopy_mean <- c()
canopy_sd <- c()
slope_mean <- c()
slope_median <- c()
slope_sd <- c()

gamma_shape <- c()
gamma_scale <- c()
vm_kappa <- c()

straightness <- c()
cum_dist <- c()
tot_dist <- c()
msd <- c()
intensity_use <- c()
sinuosity <- c()
tac <- c()

residence_time <- c() # residence time in hours

hr_area_50 <- c()
hr_area_75 <- c()
hr_area_95 <- c()
hr_monthly_overlap_ba <- c()
hr_monthly_overlap_vi <- c()
```

```r
tic()

for(k in 1:length(sim_data_traj_ids)) {

  sim_data_traj <- sim_data_btime %>% filter(traj == sim_data_traj_ids[k])

  # create a template raster
  xmin <- min(sim_data_traj$x2_) - buffer
  xmax <- max(sim_data_traj$x2_) + buffer
  ymin <- min(sim_data_traj$y2_) - buffer
  ymax <- max(sim_data_traj$y2_) + buffer
  template_raster <- rast(xmin = xmin, xmax = xmax,
                          ymin = ymin, ymax = ymax,
                          res = res, crs = crs("epsg:3112"))

  id[k] <- sim_data_traj$id[1]

  step_length_median[k] <- median(sim_data_traj$sl_)
  step_length_mean[k] <- mean(sim_data_traj$sl_)
  ndvi_mean[k] <- mean(sim_data_traj$ndvi_dry)
  ndvi_median[k] <- median(sim_data_traj$ndvi_dry)
  ndvi_sd[k] <- sd(sim_data_traj$ndvi_dry)
  herby_mean[k] <- mean(sim_data_traj$veg_herby)
  herby_sd[k] <- sd(sim_data_traj$veg_herby)
  canopy_mean[k] <- mean(sim_data_traj$canopy_cover/100)
  canopy_sd[k] <- sd(sim_data_traj$canopy_cover/100)
  slope_mean[k] <- mean(sim_data_traj$slope_raster)
  slope_median[k] <- median(sim_data_traj$slope_raster)
  slope_sd[k] <- sd(sim_data_traj$slope_raster)

  gamma_fit <- fit_distr(sim_data_traj$sl_, "gamma")
  gamma_shape[k] <- gamma_fit$params$shape
  gamma_scale[k] <- gamma_fit$params$scale

  vM_fit <- fit_distr(sim_data_traj$sl_, "vonmises")
  vm_kappa[k] <- vM_fit$params$kappa


  straightness[k] <- amt::straightness(sim_data_traj)
  cum_dist[k] <- amt::cum_dist(sim_data_traj)
  tot_dist[k] <- amt::tot_dist(sim_data_traj)
  msd[k] <- amt::msd(sim_data_traj)
  intensity_use[k] <- amt::intensity_use(sim_data_traj)
  sinuosity[k] <- amt::sinuosity(sim_data_traj)
  tac[k] <- amt::tac(sim_data_traj)

  residence_time[k] <- mean(sim_RT[[k]][,2], na.rm = TRUE)/60/60

  sim_hr_kde <- hr_kde(sim_data_traj, trast = template_raster,
                       levels = c(0.5, 0.75, 0.95))
  sim_hr_kde_area <- hr_area(sim_hr_kde)
```

```
  hr_area_50[k] = sim_hr_kde_area[which(sim_hr_kde_area$level == 0.5),]$area/1e6
  hr_area_75[k] = sim_hr_kde_area[which(sim_hr_kde_area$level == 0.75),]$area/1e6
  hr_area_95[k] = sim_hr_kde_area[which(sim_hr_kde_area$level == 0.95),]$area/1e6

  # there are three full months of data, so we can calculate the overlap
  # between monthly home ranges for these three months

  # kde 95% home ranges for each month
  sim_hr_month8 <- hr_kde(sim_data_traj %>% filter(month == 8),
                          trast = template_raster, levels = 0.95)
  sim_hr_month9 <- hr_kde(sim_data_traj %>% filter(month == 9),
                          trast = template_raster, levels = 0.95)
  sim_hr_month10 <- hr_kde(sim_data_traj %>% filter(month == 10),
                           trast = template_raster, levels = 0.95)

  # calculate the overlap between monthly KDE home ranges and take the average
  sim_hr_overlap_ba_8_9 <- hr_overlap(sim_hr_month8, sim_hr_month9, type = "ba")
  sim_hr_overlap_ba_9_10 <- hr_overlap(sim_hr_month9, sim_hr_month10, type = "ba")
  hr_monthly_overlap_ba[k] <- mean(c(sim_hr_overlap_ba_8_9$overlap,
                                     sim_hr_overlap_ba_9_10$overlap))

  sim_hr_overlap_vi_8_9 <- hr_overlap(sim_hr_month8, sim_hr_month9, type = "vi")
  sim_hr_overlap_vi_9_10 <- hr_overlap(sim_hr_month9, sim_hr_month10, type = "vi")
  hr_monthly_overlap_vi[k] <- mean(c(sim_hr_overlap_vi_8_9$overlap,
                                     sim_hr_overlap_vi_9_10$overlap))

}
```

```
## Steps with length 0 are present. This will lead to an error when fitting a gamma distribution. 0 step
## Steps with length 0 are present. This will lead to an error when fitting a gamma distribution. 0 step
## Steps with length 0 are present. This will lead to an error when fitting a gamma distribution. 0 step
```

```
toc()
```

```
## 901.75 sec elapsed
```

Create a data frame that has traj, id, and all the summaries

```
 sim_summary_df <- data.frame(

  traj = sim_data_traj_ids,
  id = as.numeric(id),
  data = "sim",
  sim = "daily",

  step_length_median =  step_length_median,
  step_length_mean =  step_length_mean,
  ndvi_mean =  ndvi_mean,
  ndvi_median =  ndvi_median,
  ndvi_sd =  ndvi_sd,
  herby_mean =  herby_mean,
  herby_sd =  herby_sd,
  canopy_mean =  canopy_mean,
  canopy_sd =  canopy_sd,
```

```
  slope_mean =  slope_mean,
  slope_median =  slope_median,
  slope_sd =  slope_sd,

  gamma_shape =  gamma_shape,
  gamma_scale =  gamma_scale,
  vm_kappa =  vm_kappa,

  straightness =  straightness,
  cum_dist =  cum_dist,
  tot_dist =  tot_dist,
  msd =  msd,
  intensity_use =  intensity_use,
  sinuosity =  sinuosity,
  tac =  tac,
  residence_time =  residence_time,

  hr_area_50 = hr_area_50,
  hr_area_75 = hr_area_75,
  hr_area_95 = hr_area_95,
  hr_monthly_overlap_ba = hr_monthly_overlap_ba,
  hr_monthly_overlap_vi = hr_monthly_overlap_vi)
```

Write the summary statistics to a csv file

```
write_csv(sim_summary_df,
          paste0("outputs/sim_2p_memALL_daily_summary_statistics_df_", Sys.Date(), ".csv"))
```

Session info

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_New Zealand.utf8  LC_CTYPE=English_New Zealand.utf8    LC_MONETARY=English_Ne
## [4] LC_NUMERIC=C                         LC_TIME=English_New Zealand.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] Rfast_2.0.7         RcppZiggurat_0.1.6  formatR_1.14        scales_1.2.1        glmmTMB_1.1.8
##  [6] clogitL1_1.5        Rcpp_1.0.10         ecospat_3.5         TwoStepCLogit_1.2.5 survival_3.5-5
## [11] viridis_0.6.2       viridisLite_0.4.2   matrixStats_1.0.0   patchwork_1.1.2     ggpubr_0.6.0
## [16] adehabitatHR_0.4.21 adehabitatLT_0.3.27 CircStats_0.2-6     boot_1.3-28.1       MASS_7.3-59
## [21] adehabitatMA_0.3.16 ade4_1.7-22         sp_1.6-0            ks_1.14.0           beepr_1.3
## [26] tictoc_1.2          terra_1.7-23        amt_0.2.1.0         lubridate_1.9.2     forcats_1.0.0
## [31] stringr_1.5.0       dplyr_1.1.2         purrr_1.0.1         readr_2.1.4         tidyr_1.3.0
```

```
## [36] tibble_3.2.1        ggplot2_3.4.2       tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##   [1] backports_1.4.1        Hmisc_5.0-1            systemfonts_1.0.4      plyr_1.8.8             TM
##   [6] splines_4.2.1          TH.data_1.1-2          digest_0.6.31          foreach_1.5.2          htm
##  [11] earth_5.3.2            fansi_1.0.4            magrittr_2.0.3         checkmate_2.1.0        clu
##  [16] tzdb_0.3.0             vroom_1.6.1            sandwich_3.0-2         timechange_0.2.0       col
##  [21] textshaping_0.3.6      rbibutils_2.2.13       xfun_0.39              crayon_1.5.2           jso
##  [26] lme4_1.1-32            zoo_1.8-12             iterators_1.0.14       ape_5.7-1              glu
##  [31] PresenceAbsence_1.1.11 gtable_0.3.3           emmeans_1.8.5          car_3.1-2              abi
##  [36] mvtnorm_1.1-3          DBI_1.1.3              rstatix_0.7.2          isoband_0.2.7          pla
##  [41] xtable_1.8-4           htmlTable_2.4.1        units_0.8-1            foreign_0.8-84         bit
##  [46] proxy_0.4-27           mclust_6.0.0           Formula_1.2-5          htmlwidgets_1.6.2      RCo
##  [51] nabor_0.5.0            pkgconfig_2.0.3        reshape_0.8.9          farver_2.1.1           nnc
##  [56] sass_0.4.5             utf8_1.2.3             tidyselect_1.2.0       labeling_0.4.2         rla
##  [61] reshape2_1.4.4         munsell_0.5.0          TeachingDemos_2.12     tools_4.2.1            cac
##  [66] cli_3.6.1              generics_0.1.3         audio_0.1-10           broom_1.0.4            eva
##  [71] fastmap_1.1.1          yaml_2.3.7             ragg_1.2.5             maxnet_0.1.4           kni
##  [76] bit64_4.0.5            fitdistrplus_1.1-8     randomForest_4.7-1.1   nlme_3.1-162           pra
##  [81] biomod2_4.2-2          compiler_4.2.1         rstudioapi_0.14        e1071_1.7-13           ggs
##  [86] bslib_0.4.2            stringi_1.7.12         plotmo_3.6.2           highr_0.10             lat
##  [91] poibin_1.5             circular_0.4-95        Matrix_1.6-5           nloptr_2.0.3           cla
##  [96] permute_0.9-7          vegan_2.6-4            gbm_2.1.8.1            vctrs_0.6.2            pil
## [101] lifecycle_1.0.3        Rdpack_2.4             jquerylib_0.1.4        estimability_1.4.1     cow
## [106] data.table_1.14.8      raster_3.6-20          R6_2.5.1               KernSmooth_2.23-20     gri
## [111] codetools_0.2-19       gtools_3.9.4           withr_2.5.0            multcomp_1.4-23        mgc
## [116] parallel_4.2.1         hms_1.1.3              grid_4.2.1             rpart_4.1.19           coo
## [121] minqa_1.2.5            class_7.3-21           rmarkdown_2.21         carData_3.0-5          mda
## [126] sf_1.0-12              pROC_1.18.0            numDeriv_2016.8-1.1    base64enc_0.1-3        tin
```