Step generation for fitting water buffalo step selection models

Scott Forrest

2024-02-29

Here we are generating random steps for fitting step selection models, and plotting the outputs of the step generation. We are using a Gamma distribution for the step lengths and a von Mises distribution for the turning angles.

Setup packages

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

Import data and clean

```
## # A tibble: 6 x 4
##   id    time                 lon   lat
##   <chr> <dttm>             <dbl> <dbl>
## 1 2005  2018-07-25 10:04:02  134. -12.3
## 2 2005  2018-07-25 11:04:23  134. -12.3
## 3 2005  2018-07-25 12:04:39  134. -12.3
## 4 2005  2018-07-25 13:04:17  134. -12.3
## 5 2005  2018-07-25 14:04:39  134. -12.3
## 6 2005  2018-07-25 15:04:27  134. -12.3
```

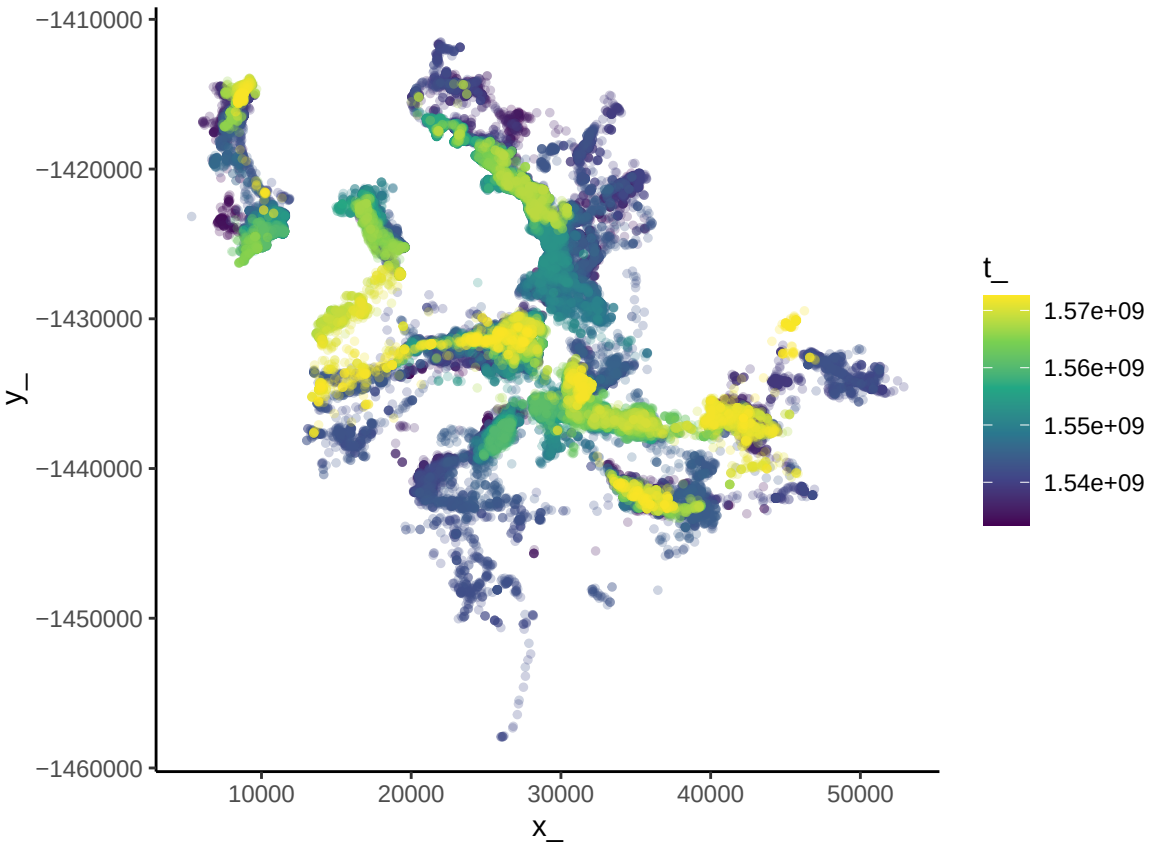
```
## [1] "Australia/Queensland"
```

Setup trajectory

Use the `amt` package to create a trajectory object from the cleaned data.

```
buffalo_all <- buffalo_clean %>% mk_track(id = id,
                                          lon,
                                          lat,
                                          time,
                                          all_cols = T,
                                          crs = 4326) %>%
  transform_coords(crs_to = 3112, crs_from = 4326) # Transformation to GDA94 /
# Geoscience Australia Lambert (https://epsg.io/3112)

# plot the data spatially coloured by time
buffalo_all %>%
  ggplot(aes(x = x_, y = y_, colour = t_)) +
  geom_point(alpha = 0.25, size = 1) + # colour = "black",
  coord_fixed() +
```

```
  scale_colour_viridis_c() +
  theme_classic()
```



Prepare environmental covariates

To determine required extent of raster to crop environmental variables to (with a buffer to accommodate the random steps)

```
min(buffalo_all$x_)
```

```
## [1] 5337.494
```

```
max(buffalo_all$x_)
```

```
## [1] 52901.66
```

```
min(buffalo_all$y_)
```

```
## [1] -1457924
```

```
max(buffalo_all$y_)
```

```
## [1] -1411517
```

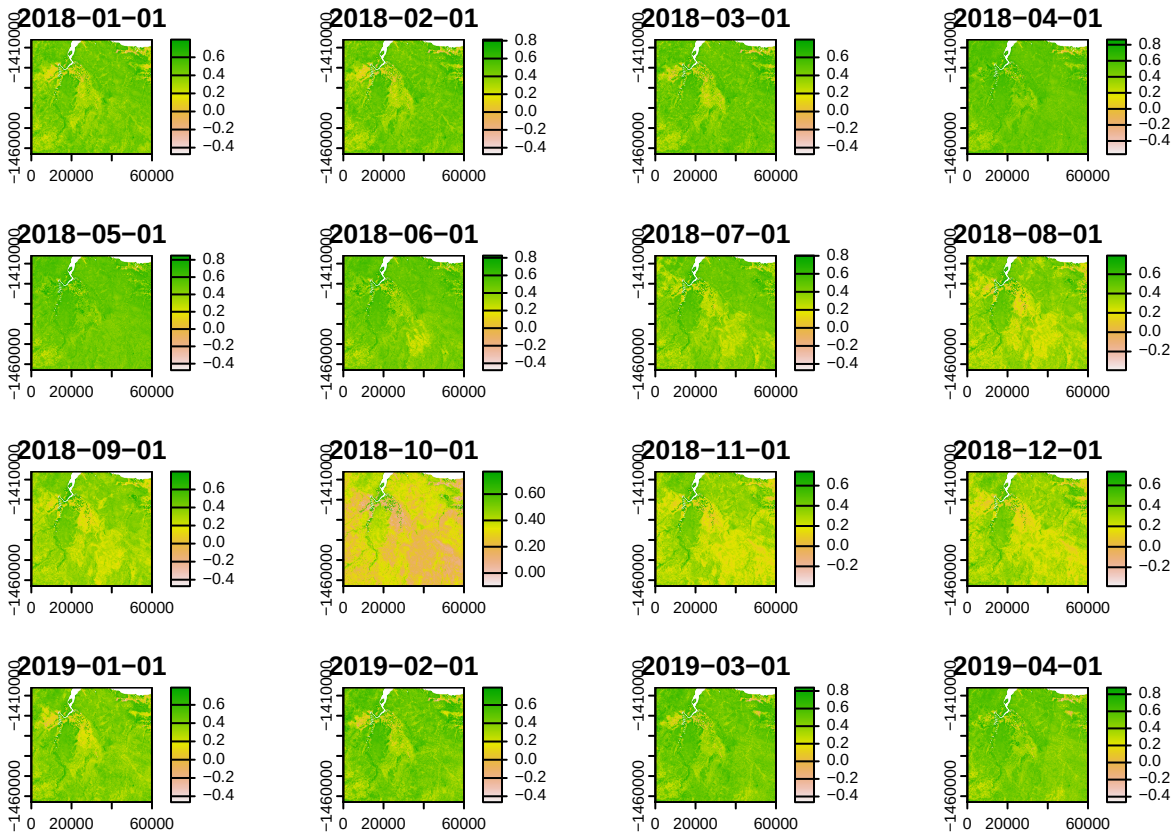
```
template_raster_cropped <- terra::rast(xmin = 0,
                                       xmax = 60000,
                                       ymin = -1463000,
                                       ymax = -1406000,
                                       resolution = 25,
```

```
                                crs = "epsg:3112")

template_wgs84 <- project(template_raster_cropped, "epsg:4326")
terra::ext(template_wgs84)
```

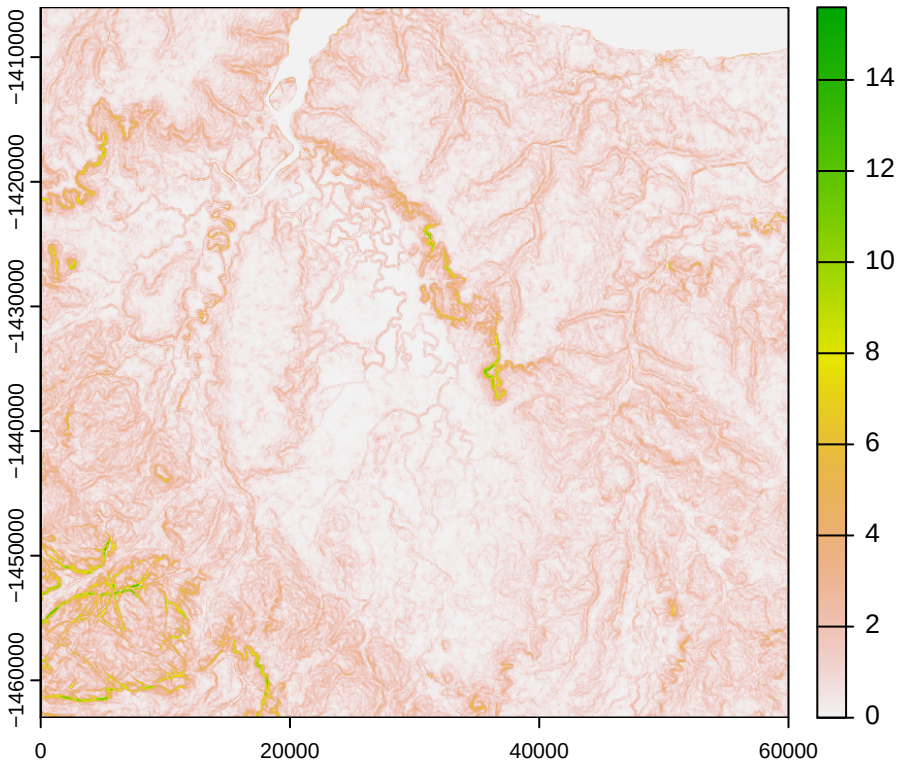
```
## SpatExtent : 134.000008561276, 134.541785217172, -12.5424223864104, -12.0361830606666 (xmin, xmax, y
```

Reading in the environmental covariates

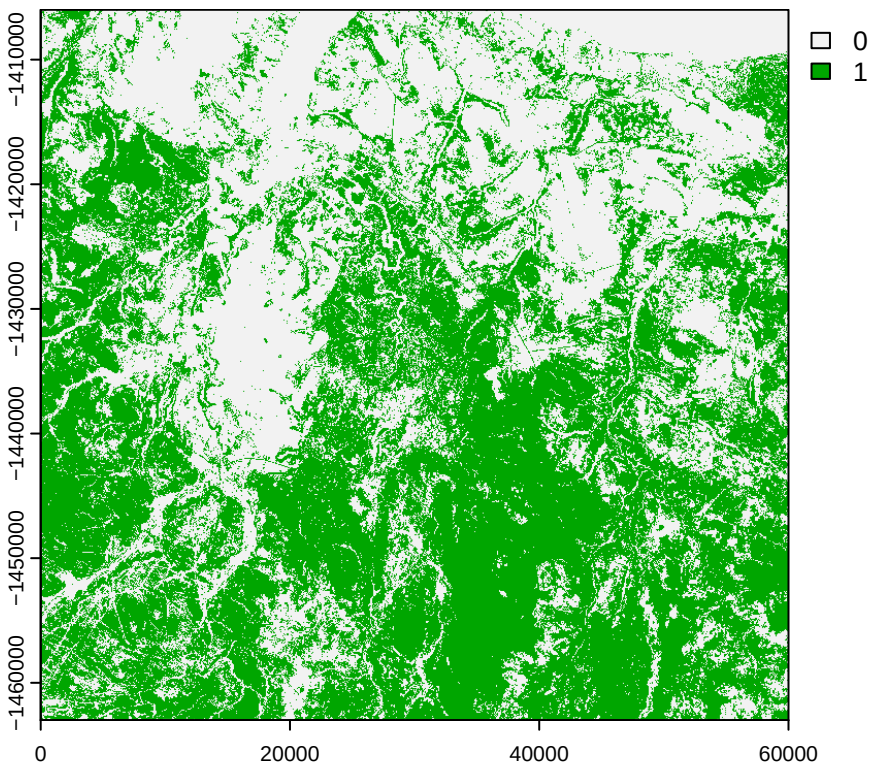
```
ndvi_projected <-
  rast("mapping/cropped rasters/ndvi_GEE_projected_watermask20230207.tif")
slope <- rast("mapping/cropped rasters/slope_raster.tif")
veg_herby <- rast("mapping/cropped rasters/veg_herby.tif")
canopy_cover <- rast("mapping/cropped rasters/canopy_cover.tif")

# change the names (these will become the column names when extracting
# covariate values at the used and random steps)
names(ndvi_projected) <- rep("ndvi", terra::nlyr(ndvi_projected))
names(slope) <- "slope"
names(veg_herby) <- "veg_herby"
names(canopy_cover) <- "canopy_cover"

# plot the rasters
plot(ndvi_projected)
```



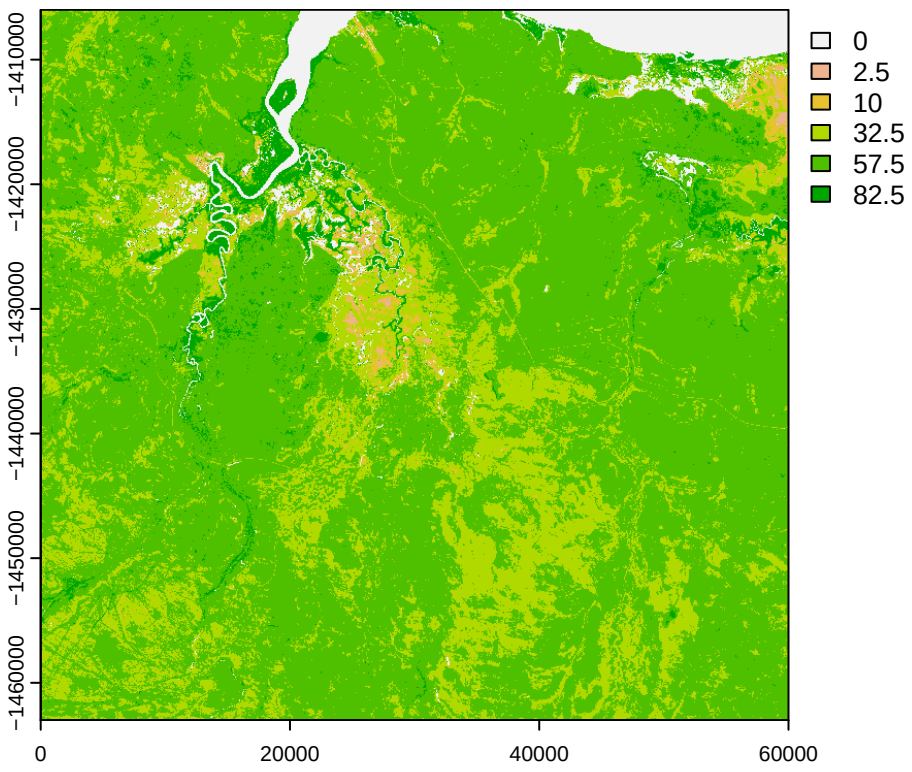
```
plot(slope)
```




```
plot(veg_herby)
```




```
plot(canopy_cover)
```



Creating a step object (by bursts)

```
# nest the data by individual
buffalo_all_nested <- buffalo_all %>% arrange(id) %>% nest(data = -"id")

buffalo_all_nested_steps_by_burst <- buffalo_all_nested %>%
  mutate(steps = map(data, function(x)
    x %>% track_resample(rate = hours(1), tolerance = minutes(10)) %>%
      # to filter out bursts with less than 3 locations
      # amt::filter_min_n_burst(min_n = 3) %>%
      steps_by_burst()))

# unnest the data after creating 'steps' objects
buffalo_all_steps_by_burst <- buffalo_all_nested_steps_by_burst %>%
  amt::select(id, steps) %>%
  amt::unnest(cols = steps)

buffalo_all_steps_by_burst <- buffalo_all_steps_by_burst %>%
  mutate(t2_rounded = round_date(t2_, "hour"), # round the time to the nearest hour
         hour_t2 = ifelse(hour(t2_rounded) == 0, 24, hour(t2_rounded))) # change the 0 hour to 24

head(buffalo_all_steps_by_burst, 10)
```

```
## # A tibble: 10 x 14
##    id    burst_    x1_    x2_       y1_       y2_    sl_ direction_p     ta_
##  * <chr>  <dbl>  <dbl>  <dbl>     <dbl>     <dbl>  <dbl>       <dbl>   <dbl>
```

```
##  1 2005       1 41940. 41968. -1435877. -1435673. 206.         1.43  NA
##  2 2005       1 41968. 41921. -1435673. -1435656.  50.7        2.80   1.37
##  3 2005       1 41921. 41778. -1435656. -1435602. 152.         2.78  -0.0214
##  4 2005       1 41778. 41840. -1435602. -1435637.  70.7       -0.507  2.99
##  5 2005       1 41840. 41655. -1435637. -1435606. 188.         2.98  -2.80
##  6 2005       1 41655. 41618. -1435606. -1435610.  37.1       -3.02   0.285
##  7 2005       1 41618. 41687. -1435610. -1436127. 522.        -1.44   1.58
##  8 2005       1 41687. 41683. -1436127. -1436120.   8.13       2.10  -2.75
##  9 2005       1 41683. 41674. -1436120. -1436120.   9.75      -3.13   1.05
## 10 2005       1 41674. 41424. -1436120. -1435940. 308.         2.52  -0.635
## # i 5 more variables: t1_ <dttm>, t2_ <dttm>, dt_ <drtn>, t2_rounded <dttm>,
## #   hour_t2 <dbl>
```

Fitting step length and turning angle distributions

Fitting exponential and von Mises distributions to the steps of ALL individuals (only one Gamma and one von Mises distribution for the whole population). This should be done when fitting a hierarchical model to update the 'population' parameters, but also makes it straightforward to update after model fitting to each individual separately.

```
# fitting step length and turning angle distributions to all locations
gamma_dist <- fit_distr(buffalo_all_steps_by_burst$sl_, "gamma")
vonmises_dist <- fit_distr(buffalo_all_steps_by_burst$ta_, "vonmises")

# checking parameters - which can then be saved to update movement parameters
# after fitting the step selection model
gamma_dist$params$shape
```

```
## [1] 0.438167
```

```
gamma_dist$params$scale
```

```
## [1] 534.3507
```

```
vonmises_dist$params$kappa
```

```
## [1] 0.1848126
```

```
vonmises_dist$params$mu
```

```
## Circular Data:
## Type = angles
## Units = radians
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
## [1] 0
```

For some reason, the `random_steps` function does not work when using the `bursted_steps_xyt` class. I'm not sure why (the error is `Error in bursts[[i]] : subscript out of bounds`), but it works when that class label is removed, and appears to sample random steps correctly. For taxa that have irregular fixes and many bursts, this may be worth exploring in more detail.

```
class(buffalo_all_steps_by_burst)
```

```
## [1] "bursted_steps_xyt" "steps_xyt"         "steps_xy"
## [4] "tbl_df"            "tbl"               "data.frame"
```

```
class(buffalo_all_steps_by_burst) <- class(buffalo_all_steps_by_burst)[-1]
class(buffalo_all_steps_by_burst)
```

```
## [1] "steps_xyt"  "steps_xy"   "tbl_df"     "tbl"        "data.frame"
```

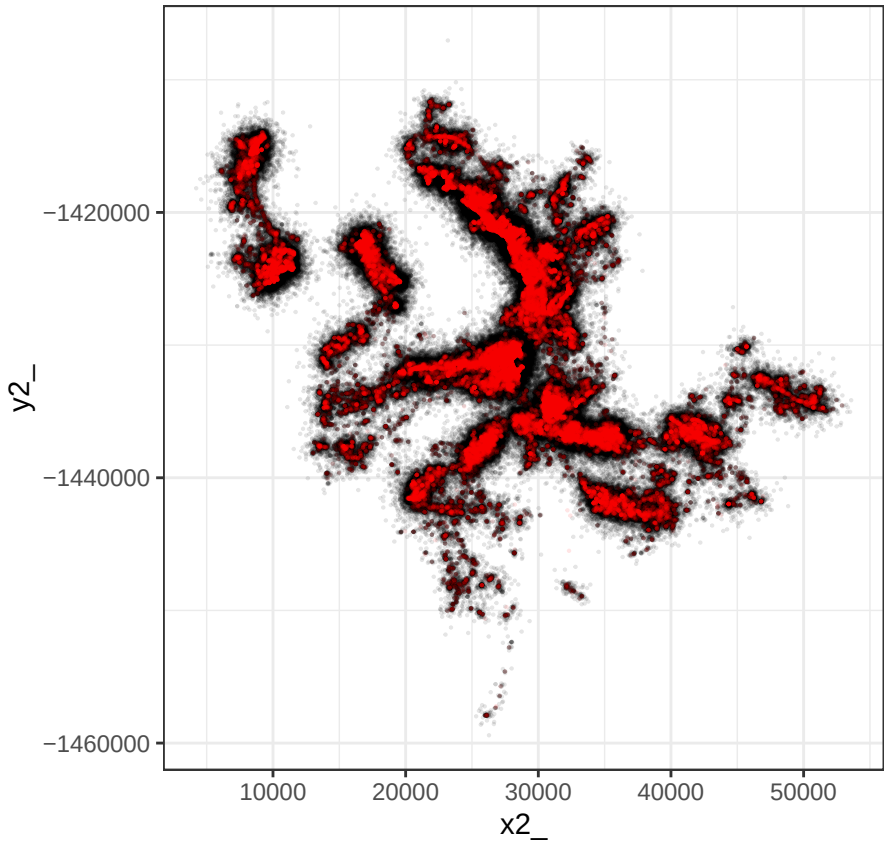
```
tic()
buffalo_parametric_popn_GvM <- buffalo_all_steps_by_burst %>%
  random_steps(n_control = 10,
               sl_distr = gamma_dist,
               ta_distr = vonmises_dist) %>%
  mutate(y = as.numeric(case_))
toc()
```

```
## 109.31 sec elapsed
```

Plotting the random step distributions

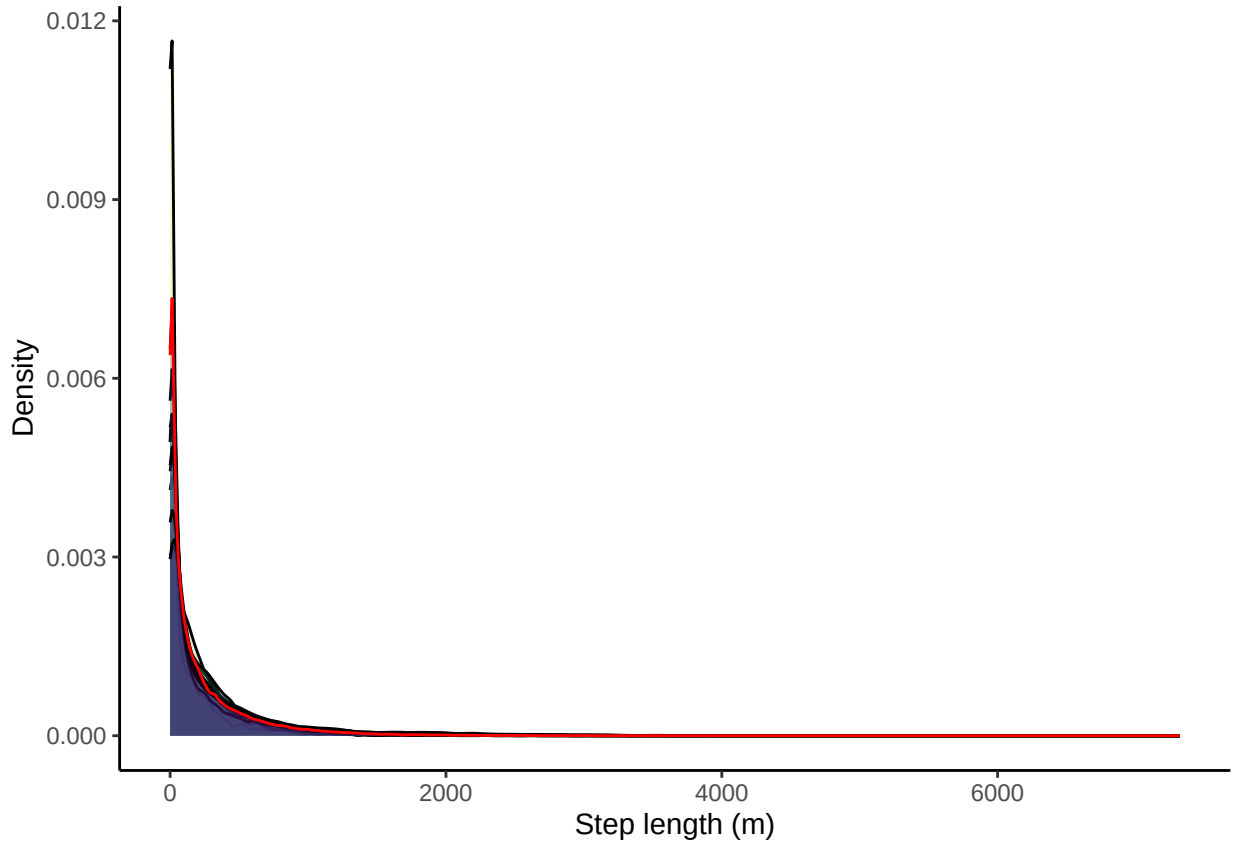
Spatially plot the used and random steps, with the used steps in red.

```
buffalo_parametric_popn_GvM %>% ggplot() +
  geom_point(data = . %>% filter(y == 0), aes(x = x2_, y = y2_),
             colour = "black", size = 0.1, alpha = 0.1) +
  geom_point(data = . %>% filter(y == 1), aes(x = x2_, y = y2_),
             colour = "red", size = 0.1, alpha = 0.1) +
  coord_equal() +
  theme_bw()
```



Gamma distribution of the step lengths

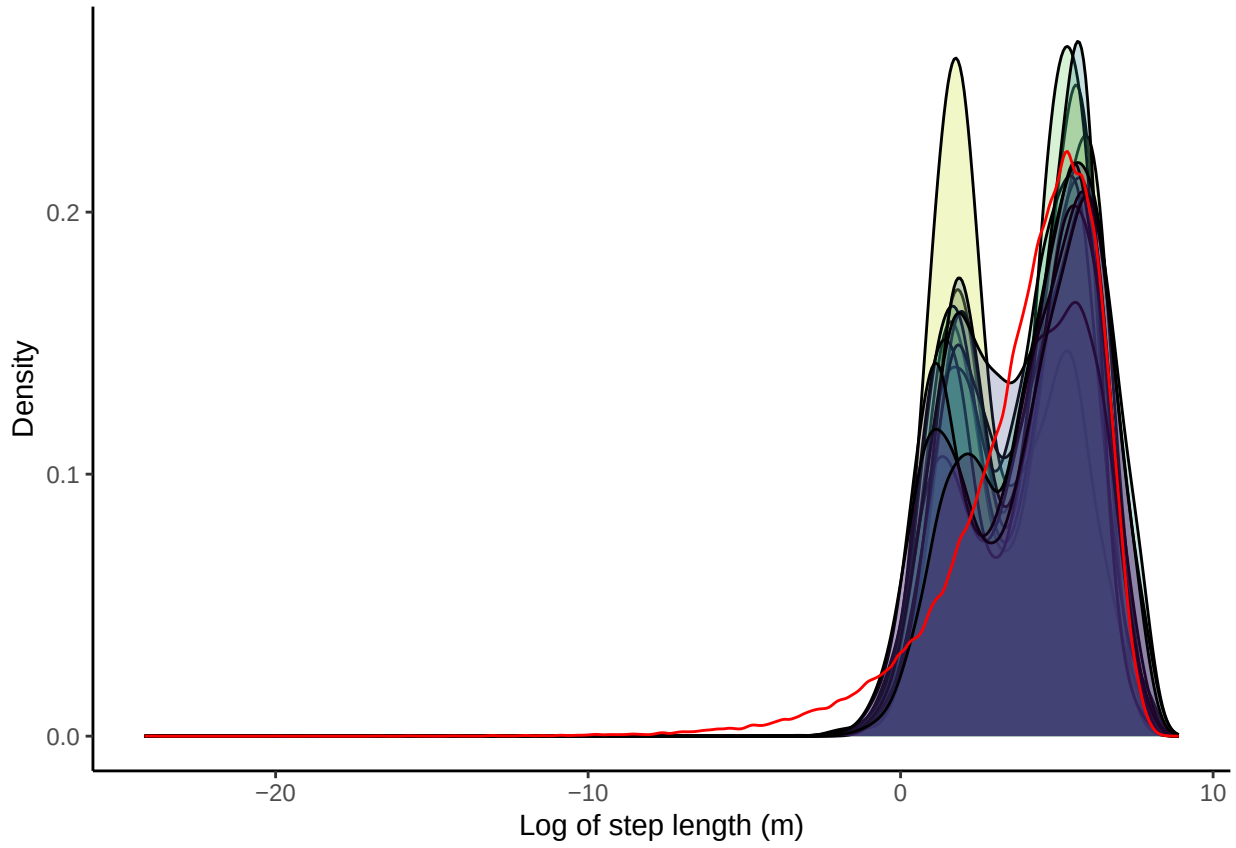
```
buffalo_parametric_popn_GvM %>% ggplot() +
  geom_density(data = . %>% filter(y == 1), aes(x = sl_, fill = factor(id)),
               alpha = 0.25) +
  geom_density(data = . %>% filter(y == 0), aes(x = sl_), colour = "red") +
  scale_y_continuous("Density") +
  scale_x_continuous("Step length (m)") +
  scale_fill_viridis_d(direction = -1) +
  theme_classic() +
  theme(legend.position = "none")
```




```
ggsave(paste0("outputs/plots/step_length_density_", Sys.Date(), ".pdf"),
       width=150, height=90, units="mm", dpi = 300)
```

When log-transforming the step length, it is clear that there is a bimodal pattern to the step lengths, which is not captured by the Gamma distribution of the random steps.

```
buffalo_parametric_popn_GvM %>% ggplot() +
  geom_density(data = . %>% filter(y == 1), aes(x = log(sl_), fill = factor(id)),
               alpha = 0.25) +
  geom_density(data = . %>% filter(y == 0), aes(x = log(sl_)), colour = "red") +
  scale_y_continuous("Density") +
  scale_x_continuous("Log of step length (m)") +
  scale_fill_viridis_d(direction = -1) +
  theme_classic() +
  theme(legend.position = "none")
```

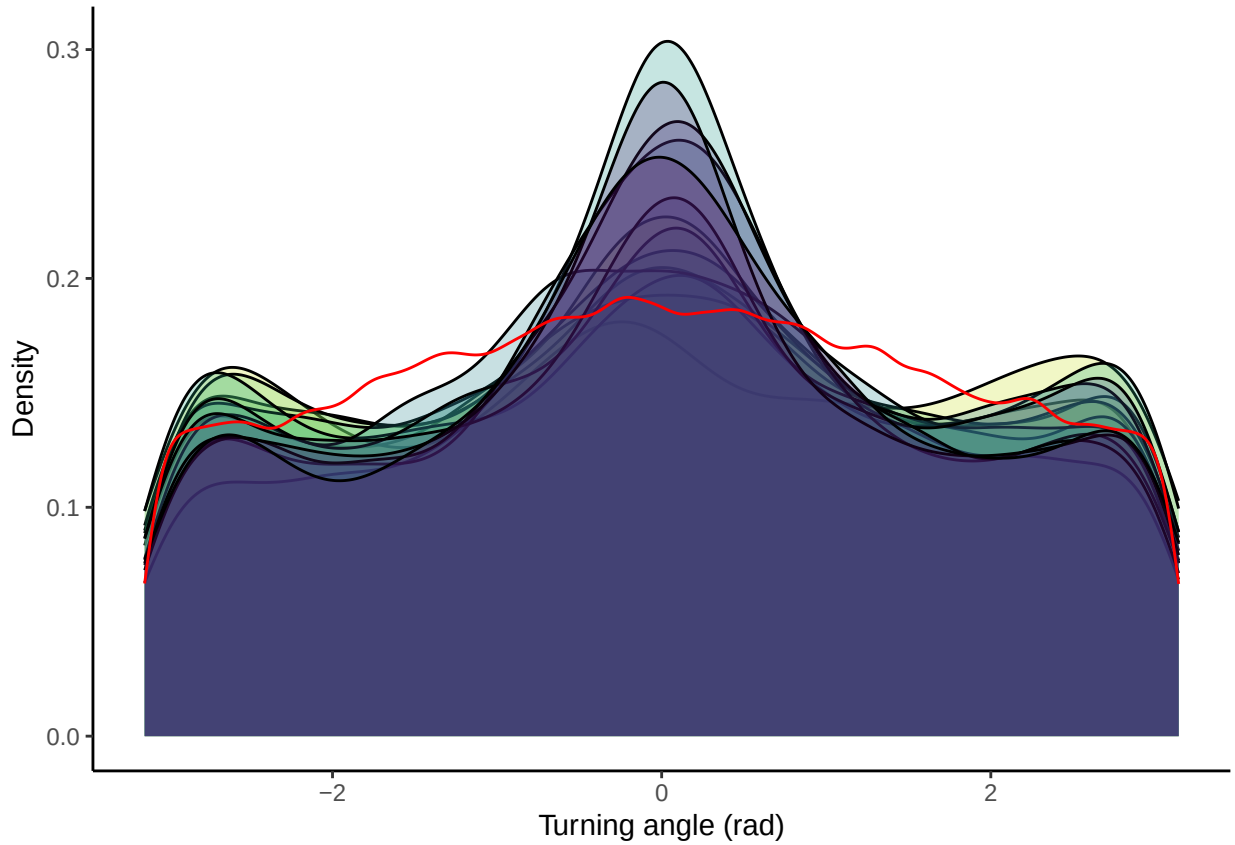



```
ggsave(paste0("outputs/plots/step_length_density_log_", Sys.Date(), ".pdf"),
       width=150, height=90, units="mm", dpi = 300)
```

Plot the von Mises distribution of the turning angles

```
buffalo_parametric_popn_GvM %>% ggplot() +
  geom_density(data = . %>% filter(y == 1), aes(x = ta_, fill = factor(id)),
               alpha = 0.25) +
  geom_density(data = . %>% filter(y == 0), aes(x = ta_), colour = "red",
               alpha = 0.01) +
  scale_y_continuous("Density") +
  scale_x_continuous("Turning angle (rad)") +
  scale_fill_viridis_d(direction = -1) +
  theme_classic() +
  theme(legend.position = "none")
```

```
## Warning: Removed 8053 rows containing non-finite values (‘stat_density()‘).
```




```
ggsave(paste0("outputs/plots/turning angle_density_", Sys.Date(), ".pdf"),
       width=150, height=90, units="mm", dpi = 300)
```

```
## Warning: Removed 8053 rows containing non-finite values (‘stat_density()‘).
```

Sample values of the environmental covariates at the end of the steps.

```
buffalo_parametric_popn_covs <- buffalo_parametric_popn_GvM %>%

  extract_covariates_var_time(ndvi_projected,
                              where = "end",
                              when = "any",
                              max_time = days(15),
                              name_covar = "ndvi_temporal") %>%
  extract_covariates(veg_herby,
                     where = "end") %>%
  extract_covariates(canopy_cover,
                     where = "end") %>%
  extract_covariates(slope,
                     where = "end") %>%

  mutate(y = as.numeric(case_),
         cos_ta_ = cos(ta_),
         log_sl_ = log(sl_))

# write the output to a csv file
```

```r
write_csv(buffalo_parametric_popn_covs,
          paste0("outputs/buffalo_parametric_popn_covs_GvM_10rs_", Sys.Date(), ".csv"))

beep(sound = 2)
```