SSF Model Fitting

Scott Forrest

2025-02-18

To compare the next-step predictions of the deepSSF models to SSF models, we need to fit some SSF models to the same data and covariates. Here we fit SSF models with and without temporal harmonics to buffalo data, which is similar to the approach in Forrest et al. (2024) except that here we are just fitting the models to the focal individual, rather than to multiple individuals.

We use the estimated parameters of the SSF models to generate next-step predictions of the SSF models in the SSF Validation script, and compare these to the next-step predictions of the deepSSF models.

Whilst we have included temporal dynamics on an daily time-scale using the harmonics, also including seasonal temporal dynamics (such that daily behaviours also change across seasons - requiring an interaction between the daily and seasonal harmonics) is difficult. We have therefore only fitted the SSF models with daily temporal dynamics.

We have also not fitted the SSF models to the Sentinel-2 data, as we have done with the deepSSF models.

Table of contents

Load packages

```
options(scipen=999)

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

Importing buffalo data

Import the buffalo data with random steps and extracted covariates that we created for the paper Forrest et al. (2024), in the script `Ecography_DynamicSSF_1_Step_generation`.

This repo can be found at: swforrest/dynamic_SSF_sims.

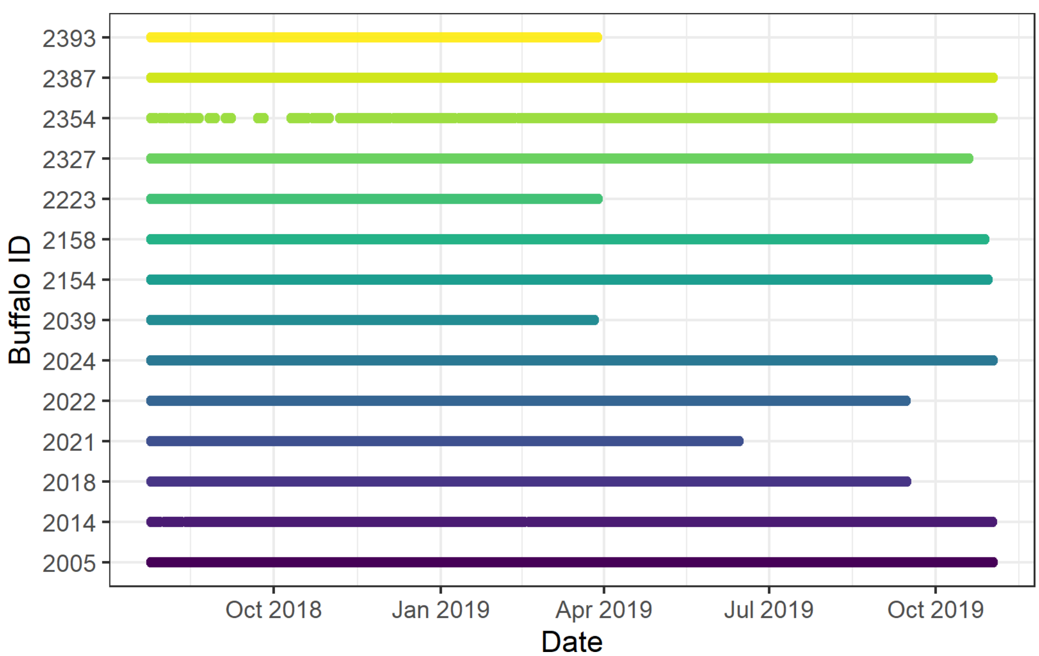
Here we create the sine and cosine terms that were interact with each of the covariates to fit temporally varying parameters.

```
buffalo_data_all <- read_csv("data/buffalo_parametric_popn_covs_GvM_10rs_2024-09-04.csv")
```

```
Rows: 1165406 Columns: 22
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl  (18): id, burst_, x1_, x2_, y1_, y2_, sl_, ta_, dt_, hour_t2, step_id_,...
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(buffalo_data_all$t1_, "hour"),
         hour_t1 = hour(t1_rounded),
         t2 = t2_,
         t2_rounded = round_date(buffalo_data_all$t2_, "hour"),
         hour_t2 = hour(t2_rounded),
         hour = hour_t2,
         yday = yday(t1_),
         year = year(t1_),
         month = month(t1_),
         sl = sl_,
         log_sl = log(sl_),
         ta = ta_,
         cos_ta = cos(ta_),
         # scale canopy cover from 0 to 1
         canopy_01 = canopy_cover/100,
         # here we create the harmonic terms for the hour of the day
         # for seasonal effects, change hour to yday (which is tau in the manuscript),
```

```
        # and 24 to 365 (which is T)
        hour_s1 = sin(2*pi*hour/24),
        hour_s2 = sin(4*pi*hour/24),
        hour_s3 = sin(6*pi*hour/24),
        hour_c1 = cos(2*pi*hour/24),
        hour_c2 = cos(4*pi*hour/24),
        hour_c3 = cos(6*pi*hour/24))

# to select a single year of data
# buffalo_data_all <- buffalo_data_all %>% filter(t1 < "2019-07-25 09:32:42 ACST")

buffalo_ids <- unique(buffalo_data_all$id)

# Timeline of buffalo data
buffalo_data_all %>% 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")
```



Fitting the models

Creating a data matrix

First we create a data matrix to be provided to the model, and then we scale and centre the full data matrix, with respect to each of the columns. That means that all variables are scaled and centred *after* the data has been split into wet and dry season data, and also after creating the quadratic and harmonic terms (when using them).

We should only include covariates in the data matrix that will be used in the model formula.

Models

- 0p = 0 pairs of harmonics
- 1p = 1 pair of harmonics
- 2p = 2 pairs of harmonics
- 3p = 3 pairs of harmonics

For the dynamic models, we start to add the harmonic terms. As we have already created the harmonic terms for the hour of the day (s1, c1, s2, etc), we just interact (multiply) these with each of the covariates, including the quadratic terms, prior to model fitting. We store the scaling and centering variables to reconstruct the natural scale coefficients.

To provide some intuition about harmonic regression we have created a walkthrough script for Forrest et al. (2024), in the script `Ecography_DynamicSSF_Walkthrough_Harmonics_and_selection_surfa` which can be found at: swforrest/dynamic_SSF_sims, that introduces harmonics and how they can be used to model temporal variation in the data. It will help provide some understand the model outputs and how we construct the temporally varying coefficients in this script.

Selecting data

```
months_wet <- c(1:4, 11:12)
buffalo_ids <- unique(buffalo_data_all$id)
focal_id <- 2005

# comment and uncomment the relevant lines to get either wet or dry season data
# buffalo_data <- buffalo_data_all %>% filter(id == focal_id & month %in% months_wet) # wet
# buffalo_data <- buffalo_data_all %>% filter(id == focal_id & !month %in% months_wet) # dry

# all data
buffalo_data <- buffalo_data_all %>% filter(id == focal_id)
```

0p

```
buffalo_data_matrix_unscaled <- buffalo_data %>% transmute(

  ndvi = ndvi_temporal,
  ndvi_sq = ndvi_temporal ^ 2,
  canopy = canopy_01,
  canopy_sq = canopy_01 ^ 2,
  slope = slope,
  herby = veg_herby,
  step_l = sl,
  log_step_l = log_sl,
  cos_turn_a = cos_ta)

buffalo_data_matrix_scaled <- scale(buffalo_data_matrix_unscaled)

# save the scaling values to recover the natural scale of the coefficients
# which is required for the simulations
# (so then environmental variables don't need to be scaled)
mean_vals <- attr(buffalo_data_matrix_scaled, "scaled:center")
sd_vals <- attr(buffalo_data_matrix_scaled, "scaled:scale")
scaling_attributes_0p <- data.frame(variable = names(buffalo_data_matrix_unscaled),
                                    mean = mean_vals, sd = sd_vals)

# add the id, step_id columns and presence/absence columns to
# the scaled data matrix for model fitting
buffalo_data_scaled_0p <- data.frame(id = buffalo_data$id,
                                     step_id = buffalo_data$step_id,
                                     y = buffalo_data$y,
                                     buffalo_data_matrix_scaled)
```

1p

```
buffalo_data_matrix_unscaled <- buffalo_data %>% transmute(

  # the 'linear' term
  ndvi = ndvi_temporal,
  # interact with the harmonic terms
  ndvi_s1 = ndvi_temporal * hour_s1,
  ndvi_c1 = ndvi_temporal * hour_c1,

  ndvi_sq = ndvi_temporal ^ 2,
  ndvi_sq_s1 = (ndvi_temporal ^ 2) * hour_s1,
```

```r
  ndvi_sq_c1 = (ndvi_temporal ^ 2) * hour_c1,

  canopy = canopy_01,
  canopy_s1 = canopy_01 * hour_s1,
  canopy_c1 = canopy_01 * hour_c1,

  canopy_sq = canopy_01 ^ 2,
  canopy_sq_s1 = (canopy_01 ^ 2) * hour_s1,
  canopy_sq_c1 = (canopy_01 ^ 2) * hour_c1,

  slope = slope,
  slope_s1 = slope * hour_s1,
  slope_c1 = slope * hour_c1,

  herby = veg_herby,
  herby_s1 = veg_herby * hour_s1,
  herby_c1 = veg_herby * hour_c1,

  step_l = sl,
  step_l_s1 = sl * hour_s1,
  step_l_c1 = sl * hour_c1,

  log_step_l = log_sl,
  log_step_l_s1 = log_sl * hour_s1,
  log_step_l_c1 = log_sl * hour_c1,

  cos_turn_a = cos_ta,
  cos_turn_a_s1 = cos_ta * hour_s1,
  cos_turn_a_c1 = cos_ta * hour_c1)

buffalo_data_matrix_scaled <- scale(buffalo_data_matrix_unscaled)

mean_vals <- attr(buffalo_data_matrix_scaled, "scaled:center")
sd_vals <- attr(buffalo_data_matrix_scaled, "scaled:scale")
scaling_attributes_1p <- data.frame(variable = names(buffalo_data_matrix_unscaled),
                                    mean = mean_vals, sd = sd_vals)

buffalo_data_scaled_1p <- data.frame(id = buffalo_data$id,
                                     step_id = buffalo_data$step_id,
                                     y = buffalo_data$y,
                                     buffalo_data_matrix_scaled)
```

2p

```
buffalo_data_matrix_unscaled <- buffalo_data %>% transmute(

  ndvi = ndvi_temporal,
  ndvi_s1 = ndvi_temporal * hour_s1,
  ndvi_s2 = ndvi_temporal * hour_s2,
  ndvi_c1 = ndvi_temporal * hour_c1,
  ndvi_c2 = ndvi_temporal * hour_c2,

  ndvi_sq = ndvi_temporal ^ 2,
  ndvi_sq_s1 = (ndvi_temporal ^ 2) * hour_s1,
  ndvi_sq_s2 = (ndvi_temporal ^ 2) * hour_s2,
  ndvi_sq_c1 = (ndvi_temporal ^ 2) * hour_c1,
  ndvi_sq_c2 = (ndvi_temporal ^ 2) * hour_c2,

  canopy = canopy_01,
  canopy_s1 = canopy_01 * hour_s1,
  canopy_s2 = canopy_01 * hour_s2,
  canopy_c1 = canopy_01 * hour_c1,
  canopy_c2 = canopy_01 * hour_c2,

  canopy_sq = canopy_01 ^ 2,
  canopy_sq_s1 = (canopy_01 ^ 2) * hour_s1,
  canopy_sq_s2 = (canopy_01 ^ 2) * hour_s2,
  canopy_sq_c1 = (canopy_01 ^ 2) * hour_c1,
  canopy_sq_c2 = (canopy_01 ^ 2) * hour_c2,

  slope = slope,
  slope_s1 = slope * hour_s1,
  slope_s2 = slope * hour_s2,
  slope_c1 = slope * hour_c1,
  slope_c2 = slope * hour_c2,

  herby = veg_herby,
  herby_s1 = veg_herby * hour_s1,
  herby_s2 = veg_herby * hour_s2,
  herby_c1 = veg_herby * hour_c1,
  herby_c2 = veg_herby * hour_c2,

  step_l = sl,
  step_l_s1 = sl * hour_s1,
  step_l_s2 = sl * hour_s2,
  step_l_c1 = sl * hour_c1,
```

```
  step_l_c2 = sl * hour_c2,

  log_step_l = log_sl,
  log_step_l_s1 = log_sl * hour_s1,
  log_step_l_s2 = log_sl * hour_s2,
  log_step_l_c1 = log_sl * hour_c1,
  log_step_l_c2 = log_sl * hour_c2,

  cos_turn_a = cos_ta,
  cos_turn_a_s1 = cos_ta * hour_s1,
  cos_turn_a_s2 = cos_ta * hour_s2,
  cos_turn_a_c1 = cos_ta * hour_c1,
  cos_turn_a_c2 = cos_ta * hour_c2)

buffalo_data_matrix_scaled <- scale(buffalo_data_matrix_unscaled)

mean_vals <- attr(buffalo_data_matrix_scaled, "scaled:center")
sd_vals <- attr(buffalo_data_matrix_scaled, "scaled:scale")
scaling_attributes_2p <- data.frame(variable = names(buffalo_data_matrix_unscaled),
                                    mean = mean_vals, sd = sd_vals)

buffalo_data_scaled_2p <- data.frame(id = buffalo_data$id,
                                     step_id = buffalo_data$step_id,
                                     y = buffalo_data$y,
                                     buffalo_data_matrix_scaled)
```

3p

```
buffalo_data_matrix_unscaled <- buffalo_data %>% transmute(

  ndvi = ndvi_temporal,
  ndvi_s1 = ndvi_temporal * hour_s1,
  ndvi_s2 = ndvi_temporal * hour_s2,
  ndvi_s3 = ndvi_temporal * hour_s3,
  ndvi_c1 = ndvi_temporal * hour_c1,
  ndvi_c2 = ndvi_temporal * hour_c2,
  ndvi_c3 = ndvi_temporal * hour_c3,

  ndvi_sq = ndvi_temporal ^ 2,
  ndvi_sq_s1 = (ndvi_temporal ^ 2) * hour_s1,
  ndvi_sq_s2 = (ndvi_temporal ^ 2) * hour_s2,
  ndvi_sq_s3 = (ndvi_temporal ^ 2) * hour_s3,
  ndvi_sq_c1 = (ndvi_temporal ^ 2) * hour_c1,
```

```
ndvi_sq_c2 = (ndvi_temporal ^ 2) * hour_c2,
ndvi_sq_c3 = (ndvi_temporal ^ 2) * hour_c3,

canopy = canopy_01,
canopy_s1 = canopy_01 * hour_s1,
canopy_s2 = canopy_01 * hour_s2,
canopy_s3 = canopy_01 * hour_s3,
canopy_c1 = canopy_01 * hour_c1,
canopy_c2 = canopy_01 * hour_c2,
canopy_c3 = canopy_01 * hour_c3,

canopy_sq = canopy_01 ^ 2,
canopy_sq_s1 = (canopy_01 ^ 2) * hour_s1,
canopy_sq_s2 = (canopy_01 ^ 2) * hour_s2,
canopy_sq_s3 = (canopy_01 ^ 2) * hour_s3,
canopy_sq_c1 = (canopy_01 ^ 2) * hour_c1,
canopy_sq_c2 = (canopy_01 ^ 2) * hour_c2,
canopy_sq_c3 = (canopy_01 ^ 2) * hour_c3,

slope = slope,
slope_s1 = slope * hour_s1,
slope_s2 = slope * hour_s2,
slope_s3 = slope * hour_s3,
slope_c1 = slope * hour_c1,
slope_c2 = slope * hour_c2,
slope_c3 = slope * hour_c3,

herby = veg_herby,
herby_s1 = veg_herby * hour_s1,
herby_s2 = veg_herby * hour_s2,
herby_s3 = veg_herby * hour_s3,
herby_c1 = veg_herby * hour_c1,
herby_c2 = veg_herby * hour_c2,
herby_c3 = veg_herby * hour_c3,

step_l = sl,
step_l_s1 = sl * hour_s1,
step_l_s2 = sl * hour_s2,
step_l_s3 = sl * hour_s3,
step_l_c1 = sl * hour_c1,
step_l_c2 = sl * hour_c2,
step_l_c3 = sl * hour_c3,

log_step_l = log_sl,
```

```
  log_step_l_s1 = log_sl * hour_s1,
  log_step_l_s2 = log_sl * hour_s2,
  log_step_l_s3 = log_sl * hour_s3,
  log_step_l_c1 = log_sl * hour_c1,
  log_step_l_c2 = log_sl * hour_c2,
  log_step_l_c3 = log_sl * hour_c3,

  cos_turn_a = cos_ta,
  cos_turn_a_s1 = cos_ta * hour_s1,
  cos_turn_a_s2 = cos_ta * hour_s2,
  cos_turn_a_s3 = cos_ta * hour_s3,
  cos_turn_a_c1 = cos_ta * hour_c1,
  cos_turn_a_c2 = cos_ta * hour_c2,
  cos_turn_a_c3 = cos_ta * hour_c3)

buffalo_data_matrix_scaled <- scale(buffalo_data_matrix_unscaled)

mean_vals <- attr(buffalo_data_matrix_scaled, "scaled:center")
sd_vals <- attr(buffalo_data_matrix_scaled, "scaled:scale")
scaling_attributes_3p <- data.frame(variable = names(buffalo_data_matrix_unscaled),
                                    mean = mean_vals, sd = sd_vals)

buffalo_data_scaled_3p <- data.frame(id = buffalo_data$id,
                                     step_id = buffalo_data$step_id,
                                     y = buffalo_data$y,
                                     buffalo_data_matrix_scaled)
```

Model formula

As we have already precomputed and scaled the covariates, quadratic terms and interactions with the harmonics, we just include each parameter as a linear predictor.

0p

```
formula_0p <- y ~

  ndvi +
  ndvi_sq +
  canopy +
  canopy_sq +
  slope +
  herby +
```

```r
  step_l +
  log_step_l +
  cos_turn_a +

  strata(step_id)
```

1p

```r
formula_1p <- y ~

  ndvi +
  ndvi_s1 +
  ndvi_c1 +

  ndvi_sq +
  ndvi_sq_s1 +
  ndvi_sq_c1 +

  canopy +
  canopy_s1 +
  canopy_c1 +

  canopy_sq +
  canopy_sq_s1 +
  canopy_sq_c1 +

  slope +
  slope_s1 +
  slope_c1 +

  herby +
  herby_s1 +
  herby_c1 +

  step_l +
  step_l_s1 +
  step_l_c1 +

  log_step_l +
  log_step_l_s1 +
  log_step_l_c1 +

  cos_turn_a +
```

```
  cos_turn_a_s1 +
  cos_turn_a_c1 +

  strata(step_id)
```

2p

```
formula_2p <- y ~

  ndvi +
  ndvi_s1 +
  ndvi_s2 +
  ndvi_c1 +
  ndvi_c2 +

  ndvi_sq +
  ndvi_sq_s1 +
  ndvi_sq_s2 +
  ndvi_sq_c1 +
  ndvi_sq_c2 +

  canopy +
  canopy_s1 +
  canopy_s2 +
  canopy_c1 +
  canopy_c2 +

  canopy_sq +
  canopy_sq_s1 +
  canopy_sq_s2 +
  canopy_sq_c1 +
  canopy_sq_c2 +

  slope +
  slope_s1 +
  slope_s2 +
  slope_c1 +
  slope_c2 +

  herby +
  herby_s1 +
  herby_s2 +
  herby_c1 +
```

```
  herby_c2 +

  step_l +
  step_l_s1 +
  step_l_s2 +
  step_l_c1 +
  step_l_c2 +

  log_step_l +
  log_step_l_s1 +
  log_step_l_s2 +
  log_step_l_c1 +
  log_step_l_c2 +

  cos_turn_a +
  cos_turn_a_s1 +
  cos_turn_a_s2 +
  cos_turn_a_c1 +
  cos_turn_a_c2 +

  strata(step_id)
```

3p

```
formula_3p <- y ~

  ndvi +
  ndvi_s1 +
  ndvi_s2 +
  ndvi_s3 +
  ndvi_c1 +
  ndvi_c2 +
  ndvi_c3 +

  ndvi_sq +
  ndvi_sq_s1 +
  ndvi_sq_s2 +
  ndvi_sq_s3 +
  ndvi_sq_c1 +
  ndvi_sq_c2 +
  ndvi_sq_c3 +

  canopy +
```

```
canopy_s1 +
canopy_s2 +
canopy_s3 +
canopy_c1 +
canopy_c2 +
canopy_c3 +

canopy_sq +
canopy_sq_s1 +
canopy_sq_s2 +
canopy_sq_s3 +
canopy_sq_c1 +
canopy_sq_c2 +
canopy_sq_c3 +

slope +
slope_s1 +
slope_s2 +
slope_s3 +
slope_c1 +
slope_c2 +
slope_c3 +

herby +
herby_s1 +
herby_s2 +
herby_s3 +
herby_c1 +
herby_c2 +
herby_c3 +

step_l +
step_l_s1 +
step_l_s2 +
step_l_s3 +
step_l_c1 +
step_l_c2 +
step_l_c3 +

log_step_l +
log_step_l_s1 +
log_step_l_s2 +
log_step_l_s3 +
log_step_l_c1 +
```

```
  log_step_l_c2 +
  log_step_l_c3 +

  cos_turn_a +
  cos_turn_a_s1 +
  cos_turn_a_s2 +
  cos_turn_a_s3 +
  cos_turn_a_c1 +
  cos_turn_a_c2 +
  cos_turn_a_c3 +

  strata(step_id)
```

Fit the model

As we have already fitted the model, we will load it here, but if the model_fit file doesn't exist, it will run the model fitting code. Be careful here that if you change the model formula, you will need to delete or rename the model_fit file to re-run the model fitting code, otherwise it will just load the previous model.

We are fitting a single model to the focal individual.

0p

```
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_0p_harms.rds"))) {

  model_0p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_0p_harms.rds"))
  print("Read existing model")

} else {

  tic()
    model_0p_harms <- fit_clogit(formula = formula_0p,
                                 data = buffalo_data_scaled_0p)
  toc()

  # save model object
  saveRDS(model_0p_harms, file = paste0("ssf_coefficients/model_id", focal_id, "_0p_harms.rd

  print("Fitted model")
  beep(sound = 2)

}
```

```
[1] "Read existing model"
```

```r
model_0p_harms
```

```
$model
Call:
survival::clogit(formula, data = data, ...)

                coef exp(coef)  se(coef)       z                  p
ndvi        0.119793  1.127263  0.054606   2.194           0.028254
ndvi_sq    -0.029336  0.971090  0.057424  -0.511           0.609444
canopy     -0.209316  0.811139  0.055978  -3.739           0.000185
canopy_sq   0.067734  1.070080  0.056884   1.191           0.233758
slope      -0.081189  0.922019  0.018447  -4.401          0.0000108
herby      -0.060009  0.941756  0.016352  -3.670           0.000243
step_l     -0.176031  0.838592  0.016867 -10.436 < 0.0000000000000002
log_step_l  0.127038  1.135461  0.015469   8.212 < 0.0000000000000002
cos_turn_a  0.001974  1.001976  0.011025   0.179           0.857924

Likelihood ratio test=282.9  on 9 df, p=< 0.00000000000000022
n= 104742, number of events= 9082
   (2574 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL

$more
NULL

attr(,"class")
[1] "fit_clogit" "list"
```

1p

```r
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_1p_harms.rds"))) {

  model_1p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_1p_harms.rds"))
  print("Read existing model")

} else {
```

```
  tic()
  model_1p_harms <- fit_clogit(formula = formula_1p,
                               data = buffalo_data_scaled_1p)
  toc()

  # save model object
  saveRDS(model_1p_harms, file = paste0("ssf_coefficients/model_id", focal_id, "_1p_harms.rd

  print("Fitted model")
  beep(sound = 2)

}
```

```
[1] "Read existing model"
```

```
model_1p_harms
```

```
$model
Call:
survival::clogit(formula, data = data, ...)

                   coef exp(coef)  se(coef)       z                  p
ndvi           0.003458  1.003464  0.065205   0.053           0.957708
ndvi_s1       -0.905497  0.404341  0.208658  -4.340 0.000014272791479765
ndvi_c1       -1.587639  0.204408  0.196747  -8.069 0.000000000000000706
ndvi_sq        0.042168  1.043069  0.066146   0.637           0.523805
ndvi_sq_s1     0.422763  1.526173  0.121607   3.476           0.000508
ndvi_sq_c1     0.894461  2.446018  0.116964   7.647 0.000000000000020526
canopy        -0.221606  0.801231  0.058306  -3.801           0.000144
canopy_s1     -0.034029  0.966543  0.166888  -0.204           0.838428
canopy_c1      0.223925  1.250977  0.169148   1.324           0.185557
canopy_sq      0.081769  1.085205  0.059131   1.383           0.166716
canopy_sq_s1   0.180573  1.197904  0.110883   1.629           0.103418
canopy_sq_c1  -0.277337  0.757799  0.112403  -2.467           0.013612
slope         -0.079070  0.923975  0.019172  -4.124 0.000037197638599163
slope_s1      -0.111915  0.894120  0.026769  -4.181 0.000029054259144576
slope_c1       0.019384  1.019573  0.027979   0.693           0.488442
herby         -0.052554  0.948803  0.017372  -3.025           0.002484
herby_s1       0.003434  1.003440  0.035854   0.096           0.923689
herby_c1       0.166075  1.180662  0.037677   4.408 0.000010438424205133
step_l        -0.236002  0.789779  0.018147 -13.005 < 0.0000000000000002
step_l_s1      0.046954  1.048074  0.021103   2.225           0.026084
step_l_c1      0.016707  1.016848  0.021392   0.781           0.434806
```

```
log_step_l     0.222075  1.248665  0.017412  12.754 < 0.0000000000000002
log_step_l_s1 -0.332569  0.717079  0.031679 -10.498 < 0.0000000000000002
log_step_l_c1 -0.467227  0.626738  0.031657 -14.759 < 0.0000000000000002
cos_turn_a     0.005601  1.005617  0.011209   0.500             0.617310
cos_turn_a_s1 -0.083722  0.919687  0.011221  -7.461 0.000000000000085936
cos_turn_a_c1 -0.097243  0.907335  0.011329  -8.583 < 0.0000000000000002

Likelihood ratio test=1136  on 27 df, p=< 0.00000000000000022
n= 104742, number of events= 9082
   (2574 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL

$more
NULL

attr(,"class")
[1] "fit_clogit" "list"
```

2p

```r
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_2p_harms.rds"))) {

  model_2p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_2p_harms.rds"))
  print("Read existing model")

} else {

  tic()
  model_2p_harms <- fit_clogit(formula = formula_2p,
                               data = buffalo_data_scaled_2p)
  toc()

  # save model object
  saveRDS(model_2p_harms, file = paste0("ssf_coefficients/model_id", focal_id, "_2p_harms.rd

  print("Fitted model")
  beep(sound = 2)

}
```

```
[1] "Read existing model"
```

```r
model_2p_harms
```

```
$model
Call:
survival::clogit(formula, data = data, ...)

                 coef exp(coef)  se(coef)       z                  p
ndvi         0.043757  1.044728  0.068423   0.640           0.522494
ndvi_s1     -0.992511  0.370645  0.205335  -4.834  0.00000134070221599
ndvi_s2      0.342154  1.407978  0.203008   1.685           0.091907
ndvi_c1     -1.612940  0.199301  0.220780  -7.306  0.00000000000027593
ndvi_c2      0.088936  1.093010  0.217183   0.409           0.682176
ndvi_sq     -0.008091  0.991942  0.069470  -0.116           0.907284
ndvi_sq_s1   0.514073  1.672089  0.120387   4.270  0.00001953181494962
ndvi_sq_s2  -0.130500  0.877657  0.120427  -1.084           0.278525
ndvi_sq_c1   0.895540  2.448658  0.130059   6.886  0.00000000000575333
ndvi_sq_c2   0.082307  1.085789  0.127867   0.644           0.519776
canopy      -0.192538  0.824863  0.059616  -3.230           0.001240
canopy_s1    0.080558  1.083892  0.172367   0.467           0.640240
canopy_s2   -0.015172  0.984942  0.168208  -0.090           0.928129
canopy_c1    0.266237  1.305045  0.177146   1.503           0.132858
canopy_c2    0.050129  1.051407  0.173408   0.289           0.772518
canopy_sq    0.058202  1.059930  0.060273   0.966           0.334221
canopy_sq_s1  0.122514  1.130335  0.114444   1.071           0.284387
canopy_sq_s2  0.104232  1.109858  0.111811   0.932           0.351223
canopy_sq_c1 -0.276427  0.758489  0.116800  -2.367           0.017948
canopy_sq_c2  0.098530  1.103547  0.114527   0.860           0.389615
slope       -0.091073  0.912951  0.020685  -4.403  0.00001068238707322
slope_s1    -0.093865  0.910406  0.026656  -3.521           0.000429
slope_s2    -0.023585  0.976691  0.028530  -0.827           0.408417
slope_c1     0.001756  1.001758  0.031056   0.057           0.954898
slope_c2    -0.029052  0.971366  0.029142  -0.997           0.318817
herby       -0.059191  0.942527  0.017900  -3.307           0.000944
herby_s1     0.002033  1.002036  0.037217   0.055           0.956428
herby_s2    -0.000974  0.999027  0.037022  -0.026           0.979011
herby_c1     0.115076  1.121959  0.040037   2.874           0.004050
herby_c2    -0.128467  0.879443  0.037886  -3.391           0.000697
step_l      -0.419477  0.657391  0.022905 -18.314 < 0.0000000000000002
step_l_s1   -0.001464  0.998537  0.019972  -0.073           0.941577
step_l_s2   -0.279437  0.756210  0.023197 -12.046 < 0.0000000000000002
step_l_c1   -0.107757  0.897845  0.028057  -3.841           0.000123
step_l_c2   -0.289807  0.748408  0.024142 -12.004 < 0.0000000000000002
```

```
log_step_l     0.288252  1.334093  0.018317  15.737 < 0.0000000000000002
log_step_l_s1 -0.374283  0.687782  0.035567 -10.523 < 0.0000000000000002
log_step_l_s2 -0.045758  0.955273  0.033065  -1.384            0.166397
log_step_l_c1 -0.372760  0.688830  0.033255 -11.209 < 0.0000000000000002
log_step_l_c2 -0.153402  0.857785  0.032811  -4.675  0.00000293525964538
cos_turn_a     0.009075  1.009116  0.011381   0.797            0.425219
cos_turn_a_s1 -0.088709  0.915112  0.011422  -7.766  0.00000000000000808
cos_turn_a_s2 -0.105611  0.899774  0.011399  -9.265 < 0.0000000000000002
cos_turn_a_c1 -0.089552  0.914341  0.011476  -7.804  0.00000000000000601
cos_turn_a_c2 -0.077023  0.925869  0.011429  -6.739  0.00000000001591447

Likelihood ratio test=2039  on 45 df, p=< 0.00000000000000022
n= 104742, number of events= 9082
   (2574 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL

$more
NULL

attr(,"class")
[1] "fit_clogit" "list"
```

3p

```
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_3p_harms.rds"))) {

  model_3p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_3p_harms.rds"))
  print("Read existing model")

} else {

  tic()
  model_3p_harms <- fit_clogit(formula = formula_3p,
                               data = buffalo_data_scaled_3p)
  toc()

  # save model object
  saveRDS(model_3p_harms, file = paste0("ssf_coefficients/model_id", focal_id, "_3p_harms.rd
```

```
  print("Fitted model")
  beep(sound = 2)

}
```

```
[1] "Read existing model"
```

```
model_3p_harms
```

```
$model
Call:
survival::clogit(formula, data = data, ...)

                 coef exp(coef)  se(coef)      z                  p
ndvi         0.053434  1.054887  0.069905  0.764           0.444642
ndvi_s1     -0.889595  0.410822  0.220413 -4.036 0.0000543614715317
ndvi_s2      0.376174  1.456700  0.210826  1.784           0.074378
ndvi_s3      0.020205  1.020410  0.221771  0.091           0.927409
ndvi_c1     -1.673562  0.187578  0.218941 -7.644 0.0000000000000210
ndvi_c2     -0.135140  0.873594  0.227080 -0.595           0.551764
ndvi_c3     -0.208753  0.811596  0.207927 -1.004           0.315391
ndvi_sq     -0.013759  0.986335  0.070977 -0.194           0.846290
ndvi_sq_s1   0.452224  1.571804  0.128359  3.523           0.000426
ndvi_sq_s2  -0.166224  0.846856  0.123974 -1.341           0.179986
ndvi_sq_s3  -0.056616  0.944957  0.130381 -0.434           0.664116
ndvi_sq_c1   0.943417  2.568745  0.129567  7.281 0.0000000000033058
ndvi_sq_c2   0.226610  1.254340  0.133381  1.699           0.089326
ndvi_sq_c3   0.014961  1.015073  0.123553  0.121           0.903620
canopy      -0.210127  0.810481  0.060329 -3.483           0.000496
canopy_s1    0.139434  1.149623  0.175581  0.794           0.427118
canopy_s2    0.041171  1.042030  0.173616  0.237           0.812552
canopy_s3    0.167108  1.181882  0.172810  0.967           0.333542
canopy_c1    0.186546  1.205080  0.177467  1.051           0.293186
canopy_c2   -0.020638  0.979573  0.178995 -0.115           0.908206
canopy_c3   -0.432610  0.648814  0.172568 -2.507           0.012180
canopy_sq    0.070688  1.073247  0.061074  1.157           0.247099
canopy_sq_s1 0.079781  1.083050  0.116564  0.684           0.493698
canopy_sq_s2 0.068549  1.070953  0.115436  0.594           0.552631
canopy_sq_s3 -0.128069 0.879793  0.114703 -1.117           0.264199
canopy_sq_c1 -0.233344 0.791881  0.117332 -1.989           0.046728
canopy_sq_c2 0.128881  1.137555  0.118059  1.092           0.274978
canopy_sq_c3 0.262403  1.300051  0.114177  2.298           0.021550
slope       -0.101180  0.903771  0.020815 -4.861 0.0000011679774923
```

```
slope_s1      -0.079426  0.923646  0.027910  -2.846             0.004430
slope_s2      -0.018933  0.981245  0.028913  -0.655             0.512585
slope_s3       0.027495  1.027877  0.028756   0.956             0.338986
slope_c1       0.004549  1.004559  0.031200   0.146             0.884082
slope_c2      -0.021925  0.978314  0.029634  -0.740             0.459388
slope_c3      -0.063772  0.938219  0.029628  -2.152             0.031365
herby         -0.055395  0.946111  0.018036  -3.071             0.002131
herby_s1      -0.002842  0.997162  0.037958  -0.075             0.940322
herby_s2      -0.011763  0.988306  0.038665  -0.304             0.760951
herby_s3      -0.057742  0.943893  0.038132  -1.514             0.129954
herby_c1       0.138502  1.148552  0.040208   3.445             0.000572
herby_c2      -0.096224  0.908260  0.039346  -2.446             0.014463
herby_c3       0.046576  1.047677  0.037659   1.237             0.216170
step_l        -0.475893  0.621330  0.023495 -20.255 < 0.0000000000000002
step_l_s1      0.082577  1.086082  0.024644   3.351             0.000806
step_l_s2     -0.235319  0.790319  0.024867  -9.463 < 0.0000000000000002
step_l_s3      0.037193  1.037893  0.024911   1.493             0.135428
step_l_c1      0.037076  1.037772  0.029939   1.238             0.215564
step_l_c2     -0.207766  0.812397  0.025781  -8.059  0.00000000000000077
step_l_c3     -0.016837  0.983304  0.024683  -0.682             0.495166
log_step_l     0.424316  1.528544  0.021121  20.090 < 0.0000000000000002
log_step_l_s1 -0.485817  0.615194  0.042313 -11.482 < 0.0000000000000002
log_step_l_s2 -0.097189  0.907385  0.036489  -2.664             0.007732
log_step_l_s3  0.577112  1.780888  0.035215  16.388 < 0.0000000000000002
log_step_l_c1 -0.559955  0.571235  0.033581 -16.675 < 0.0000000000000002
log_step_l_c2 -0.431154  0.649759  0.037566 -11.477 < 0.0000000000000002
log_step_l_c3  0.386800  1.472262  0.034641  11.166 < 0.0000000000000002
cos_turn_a     0.005726  1.005743  0.011526   0.497             0.619330
cos_turn_a_s1 -0.083038  0.920316  0.011673  -7.114  0.00000000000112854
cos_turn_a_s2 -0.099854  0.904970  0.011521  -8.667 < 0.0000000000000002
cos_turn_a_s3  0.145950  1.157139  0.011610  12.571 < 0.0000000000000002
cos_turn_a_c1 -0.101155  0.903793  0.011567  -8.745 < 0.0000000000000002
cos_turn_a_c2 -0.089038  0.914811  0.011680  -7.623  0.00000000000002471
cos_turn_a_c3  0.027900  1.028292  0.011512   2.423             0.015374

Likelihood ratio test=2898  on 63 df, p=< 0.00000000000000022
n= 104742, number of events= 9082
   (2574 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL
```

```
$more
NULL

attr(,"class")
[1] "fit_clogit" "list"
```

Check the fitted model outputs

Create a dataframe of the coefficients with the scaling attributes that we saved when creating the data matrix. We can then return the coefficients to their natural scale by dividing by the scaling factor (standard deviation).

As we can see, we have a coefficient for each covariate by itself, and then one for each of the harmonic interactions. These are the 'weights' that we played around with in the `Ecography_DynamicSSF_Walkthrough_Harmonics_and_selection_surfaces` walkthrough script in: swforrest/dynamic_SSF_sims, and we reconstruct them in exactly the same way. We also have the coefficients for the quadratic terms and the interactions with the harmonics, which we have denoted as `ndvi_sq` for instance. We will come back to these when we look at the selection surfaces.

0p

```
model_0p_harms
```

```
$model
Call:
survival::clogit(formula, data = data, ...)

                coef exp(coef)  se(coef)       z                  p
ndvi        0.119793  1.127263  0.054606   2.194           0.028254
ndvi_sq    -0.029336  0.971090  0.057424  -0.511           0.609444
canopy     -0.209316  0.811139  0.055978  -3.739           0.000185
canopy_sq   0.067734  1.070080  0.056884   1.191           0.233758
slope      -0.081189  0.922019  0.018447  -4.401          0.0000108
herby      -0.060009  0.941756  0.016352  -3.670           0.000243
step_l     -0.176031  0.838592  0.016867 -10.436 < 0.0000000000000002
log_step_l  0.127038  1.135461  0.015469   8.212 < 0.0000000000000002
cos_turn_a  0.001974  1.001976  0.011025   0.179           0.857924

Likelihood ratio test=282.9  on 9 df, p=< 0.00000000000000022
n= 104742, number of events= 9082
   (2574 observations deleted due to missingness)
```

```
$sl_
NULL

$ta_
NULL

$more
NULL

attr(,"class")
[1] "fit_clogit" "list"
```

```r
# these create massive outputs for the dynamic models so we've commented them out
# model_0p_harms$model$coefficients
# model_0p_harms$se
# model_0p_harms$vcov
# diag(model_0p_harms$D) # between cluster variance
# model_0p_harms$r.effect # individual estimates

# create a dataframe of the coefficients and their scaling attributes
coefs_clr_0p <- data.frame(coefs = names(model_0p_harms$model$coefficients),
                           value = model_0p_harms$model$coefficients)

# return coefficients to natural scale
coefs_clr_0p$scale_sd <- scaling_attributes_0p$sd
coefs_clr_0p <- coefs_clr_0p %>% mutate(value_nat = value / scale_sd)

# show the first few rows
head(coefs_clr_0p)
```

```
                coefs       value   scale_sd  value_nat
ndvi             ndvi  0.11979262 0.09970648  1.2014527
ndvi_sq       ndvi_sq -0.02933619 0.06498555 -0.4514263
canopy         canopy -0.20931554 0.15313840 -1.3668390
canopy_sq   canopy_sq  0.06773356 0.12331270  0.5492829
slope           slope -0.08118911 0.68009298 -0.1193794
herby           herby -0.06000902 0.40882526 -0.1467840
```

1p

```r
# creates a huge output due to the correlation matrix
# model_1p_harms
```

```r
# model_1p_harms
# model_1p_harms$model$coefficients
# model_1p_harms$se
# model_1p_harms$vcov
# diag(model_1p_harms$D) # between cluster variance
# model_1p_harms$r.effect # individual estimates

coefs_clr_1p <- data.frame(coefs = names(model_1p_harms$model$coefficients),
                           value = model_1p_harms$model$coefficients)

# return coefficients to natural scale
coefs_clr_1p$scale_sd <- scaling_attributes_1p$sd
coefs_clr_1p <- coefs_clr_1p %>% mutate(value_nat = value / scale_sd)

# show the first few rows
head(coefs_clr_1p)
```

```
                coefs       value   scale_sd   value_nat
ndvi             ndvi  0.00345779 0.09970648  0.03467969
ndvi_s1       ndvi_s1 -0.90549705 0.22207031 -4.07752410
ndvi_c1       ndvi_c1 -1.58763903 0.22261685 -7.13171101
ndvi_sq       ndvi_sq  0.04216766 0.06498555  0.64887747
ndvi_sq_s1 ndvi_sq_s1  0.42276326 0.08269541  5.11229399
ndvi_sq_c1 ndvi_sq_c1  0.89446133 0.08466353 10.56489576
```

2p

```r
# creates a huge output due to the correlation matrix
# model_2p_harms

# model_2p_harms
# model_2p_harms$model$coefficients
# model_2p_harms$se
# model_2p_harms$vcov
# diag(model_2p_harms$D) # between cluster variance
# model_2p_harms$r.effect # individual estimates

# creating data frame of model coefficients
coefs_clr_2p <- data.frame(coefs = names(model_2p_harms$model$coefficients),
                           value = model_2p_harms$model$coefficients)

# return coefficients to natural scale
coefs_clr_2p$scale_sd <- scaling_attributes_2p$sd
```

```r
coefs_clr_2p <- coefs_clr_2p %>% mutate(value_nat = value / scale_sd)

# show the first few rows
head(coefs_clr_2p)
```

```
          coefs       value   scale_sd  value_nat
ndvi       ndvi  0.04375683 0.09970648  0.4388565
ndvi_s1 ndvi_s1 -0.99251073 0.22207031 -4.4693535
ndvi_s2 ndvi_s2  0.34215431 0.21936365  1.5597585
ndvi_c1 ndvi_c1 -1.61294037 0.22261685 -7.2453652
ndvi_c2 ndvi_c2  0.08893551 0.22532434  0.3947000
ndvi_sq ndvi_sq -0.00809084 0.06498555 -0.1245021
```

3p

```r
# creates a huge output due to the correlation matrix
# model_3p_harms

# model_3p_harms$model$coefficients
# model_3p_harms$se
# model_3p_harms$vcov
# diag(model_3p_harms$D) # between cluster variance
# model_3p_harms$r.effect # individual estimates

# creating dataframe of coefficients
coefs_clr_3p <- data.frame(coefs = names(model_3p_harms$model$coefficients),
                           value = model_3p_harms$model$coefficients)

# return coefficients to natural scale
coefs_clr_3p$scale_sd <- scaling_attributes_3p$sd
coefs_clr_3p <- coefs_clr_3p %>% mutate(value_nat = value / scale_sd)

# show the first few rows
head(coefs_clr_3p)
```

```
          coefs       value   scale_sd   value_nat
ndvi       ndvi  0.05343376 0.09970648  0.53591063
ndvi_s1 ndvi_s1 -0.88959485 0.22207031 -4.00591521
ndvi_s2 ndvi_s2  0.37617357 0.21936365  1.71484006
ndvi_s3 ndvi_s3  0.02020461 0.22087403  0.09147573
ndvi_c1 ndvi_c1 -1.67356151 0.22261685 -7.51767674
ndvi_c2 ndvi_c2 -0.13513951 0.22532434 -0.59975548
```

Reconstruct the temporally dynamic coefficients

First we reconstruct the hourly coefficients for the model with no harmonics. This step isn't necessary as we already have the coefficients, and we have already rescaled them in the dataframe we created above. But as we are also fitting harmonic models and recover their coefficients across time, we have used the same approach here so then we can plot them together and illustrate the static/dynamic outputs of the models. It also means that we can use the same simulation code (which indexes across the hour of the day), and just change the data frame of coefficients (as it will index across the coefficients of the static model but they won't change).

We need a sequence of values that covers a full period (or the period that we want to construct the function over, which can be more or less than 1 period). The sequence can be arbitrarily finely spaced. The smaller the increment the smoother the function will be for plotting. When simulating data from the temporally dynamic coefficients, we will subset to the increment that relates to the data collection and model fitting (i.e. one hour in this case).

Essentially, the coefficients can be considered as weights of the harmonics, which combine into a single function.

Now we can reconstruct the harmonic function using the formula that we put into our model by interacting the harmonic terms with each of the covariates, for two pairs of harmonics (2p) a single covariate, let's say herbaceous vegetation (herby), this would be written down as:

$$f = \beta_{herby} + \beta_{herby\_s1} \sin\left(\frac{2\pi t}{24}\right) + \beta_{herby\_c1} \cos\left(\frac{2\pi t}{24}\right) + \beta_{herby\_s2} \sin\left(\frac{4\pi t}{24}\right) + \beta_{herby\_c2} \cos\left(\frac{4\pi t}{24}\right),$$

where we have 5 β_{herby} coefficients, one for the linear term, and one for each of the harmonic terms.

Here we use matrix multiplication to reconstruct the temporally dynamic coefficients. We provide some background in the `Ecography_DynamicSSF_Walkthrough_Harmonics_and_selection_surfaces` script.

First we create a matrix of the values of the harmonics, which is just the sin and cos terms for each harmonic, and then we can multiply this by the coefficients to get the function. When we use two pairs of harmonics we will have 5 coefficients for each covariate (linear + 2 sine and 2 cosine), so there will be 5 columns in the matrix.

For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The result will then have the same number of rows as the first matrix and the same number of columns as the second matrix.

Or in other words, if we have a 24 x 5 matrix of harmonics and a 5 x 1 matrix of coefficients, we will get a 24 x 1 matrix of the function, which corresponds to our 24 hours of the day.

0p

```
# increments are arbitrary - finer results in smoother curves
# for the simulations we will subset to the step interval
hour <- seq(0,23.9,0.1)

# create the dataframe of values of the harmonic terms over the period (here just the linear
hour_harmonics_df_0p <- data.frame("linear_term" = rep(1, length(hour)))

harmonics_scaled_df_0p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "ndvi_2" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "canopy" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "canopy_2" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "slope" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("slope", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "herby" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "sl" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "log_sl" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "cos_ta" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))))

harmonics_scaled_long_0p <- pivot_longer(harmonics_scaled_df_0p,
                                         cols = !1,
                                         names_to = "coef")
```

1p

```r
# create the dataframe of values of the harmonic terms over the period
hour_harmonics_df_1p <- data.frame("linear_term" = rep(1, length(hour)),
                                "hour_s1" = sin(2*pi*hour/24),
                                "hour_c1" = cos(2*pi*hour/24))

harmonics_scaled_df_1p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "ndvi_2" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "canopy" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "canopy_2" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "slope" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("slope", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "herby" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "sl" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "log_sl" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "cos_ta" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))))

harmonics_scaled_long_1p <- pivot_longer(harmonics_scaled_df_1p,
                                         cols = !1,
                                         names_to = "coef")
```

2p

```
# create the dataframe of values of the harmonic terms over the period
hour_harmonics_df_2p <- data.frame("linear_term" = rep(1, length(hour)),
                                   "hour_s1" = sin(2*pi*hour/24),
                                   "hour_s2" = sin(4*pi*hour/24),
                                   "hour_c1" = cos(2*pi*hour/24),
                                   "hour_c2" = cos(4*pi*hour/24))

harmonics_scaled_df_2p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "ndvi_2" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "canopy" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "canopy_2" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "slope" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("slope", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "herby" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "sl" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "log_sl" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "cos_ta" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))))

harmonics_scaled_long_2p <- pivot_longer(harmonics_scaled_df_2p, cols = !1,
                                         names_to = "coef")
```

3p

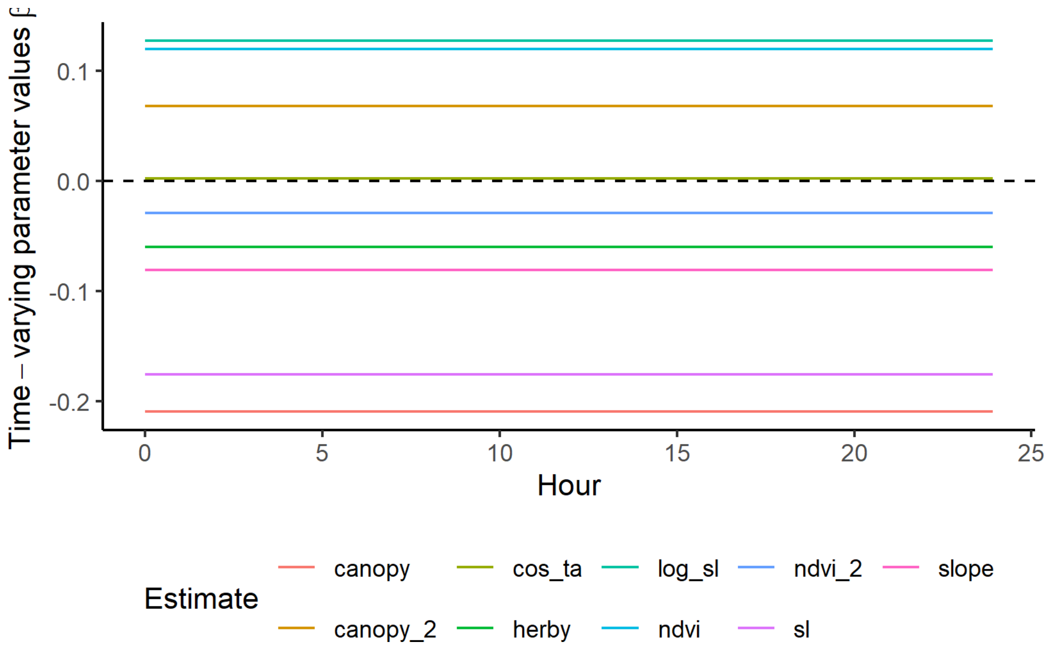
```r
# create the dataframe of values of the harmonic terms over the period
hour_harmonics_df_3p <- data.frame("linear_term" = rep(1, length(hour)),
                                   "hour_s1" = sin(2*pi*hour/24),
                                   "hour_s2" = sin(4*pi*hour/24),
                                   "hour_s3" = sin(6*pi*hour/24),
                                   "hour_c1" = cos(2*pi*hour/24),
                                   "hour_c2" = cos(4*pi*hour/24),
                                   "hour_c3" = cos(6*pi*hour/24))

harmonics_scaled_df_3p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "ndvi_2" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "canopy" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "canopy_2" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "slope" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("slope", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "herby" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "sl" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "log_sl" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "cos_ta" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))))

harmonics_scaled_long_3p <- pivot_longer(harmonics_scaled_df_3p, cols = !1,
                                         names_to = "coef")
```

Plot the results - scaled temporally dynamic coefficients

Here we show the temporally-varying coefficients across time (which are currently still scaled).

0p

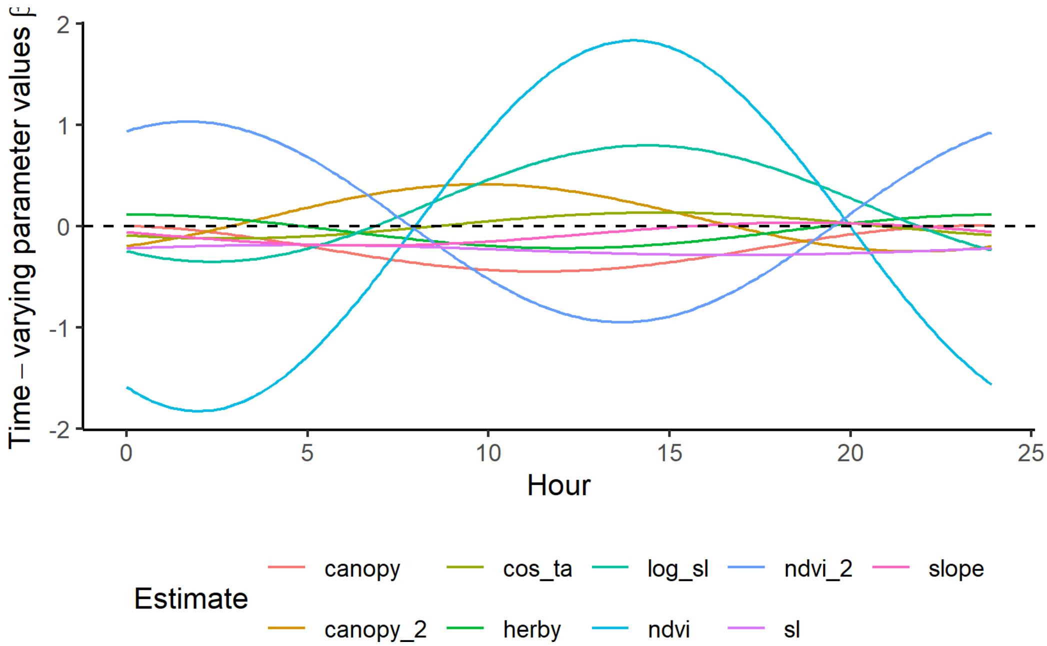
```
ggplot() +
    geom_path(data = harmonics_scaled_long_0p,
              aes(x = hour, y = value, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
    scale_x_continuous("Hour") +
    scale_color_discrete("Estimate") +
    theme_classic() +
    theme(legend.position = "bottom")
```



1p

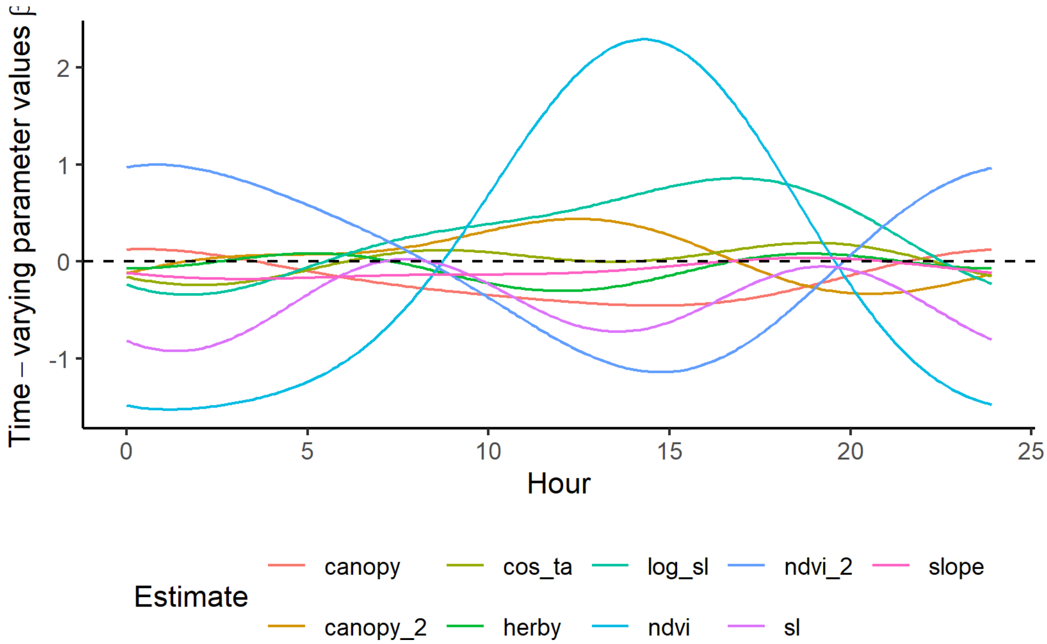
```
ggplot() +
    geom_path(data = harmonics_scaled_long_1p,
              aes(x = hour, y = value, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
```

```
    scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
    scale_x_continuous("Hour") +
    scale_color_discrete("Estimate") +
    theme_classic() +
    theme(legend.position = "bottom")
```



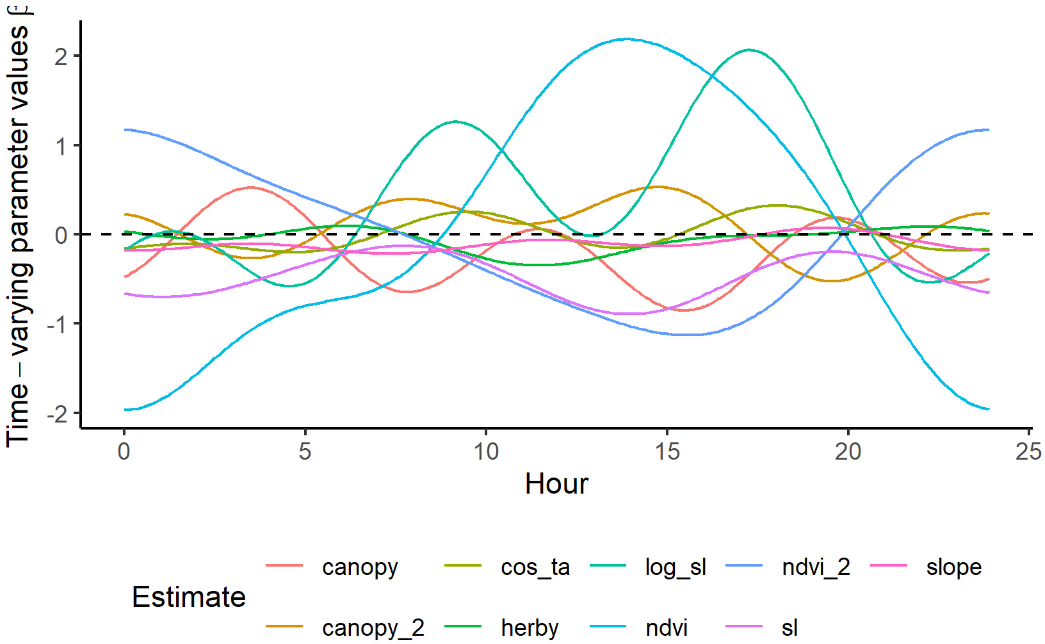
2p

```
ggplot() +
    geom_path(data = harmonics_scaled_long_2p,
              aes(x = hour, y = value, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
    scale_x_continuous("Hour") +
    scale_color_discrete("Estimate") +
    theme_classic() +
    theme(legend.position = "bottom")
```



3p

```
ggplot() +
    geom_path(data = harmonics_scaled_long_3p,
              aes(x = hour, y = value, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
    scale_x_continuous("Hour") +
    scale_color_discrete("Estimate") +
    theme_classic() +
    theme(legend.position = "bottom")
```



Reconstructing the natural-scale temporally dynamic coefficients

As we scaled the covariate values prior to fitting the models, we want to rescale the coefficients to their natural scale. This is important for the simulations, as the environmental variables will not be scaled when we simulate steps.

0p

```
harmonics_nat_df_0p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "ndvi_2" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "canopy" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "canopy_2" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "slope" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("slope", coefs) & !grepl("sq", coefs)) %>%
```

```
    pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "herby" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "sl" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "log_sl" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p))),
  "cos_ta" = as.numeric(
    coefs_clr_0p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_0p)))
```

1p

```
harmonics_nat_df_1p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "ndvi_2" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "canopy" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "canopy_2" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "slope" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("slope", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "herby" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "sl" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "log_sl" = as.numeric(
    coefs_clr_1p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))),
  "cos_ta" = as.numeric(
```

```
    coefs_clr_1p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_1p))))
```

2p

```
harmonics_nat_df_2p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "ndvi_2" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "canopy" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "canopy_2" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "slope" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("slope", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "herby" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "sl" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "log_sl" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))),
  "cos_ta" = as.numeric(
    coefs_clr_2p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_2p))))
```

3p

```
harmonics_nat_df_3p <- data.frame(
  "hour" = hour,
  "ndvi" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("ndvi", coefs) & !grepl("sq", coefs)) %>%
```

```
    pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "ndvi_2" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("ndvi_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "canopy" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("canopy", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "canopy_2" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("canopy_sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "slope" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("slope", coefs) & !grepl("sq", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "herby" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("herby", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "sl" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "log_sl" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "cos_ta" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("cos", coefs)) %>%
      pull(value_nat) %>% t() %*% t(as.matrix(hour_harmonics_df_3p)))))
```

Update the Gamma and von Mises distributions

To update the Gamma and von Mises distribution from the tentative distributions (e.g. Fieberg et al. 2021, Appendix C), we just do the calculation at each time point (for the natural-scale coefficients).

0p

```
# from the step generation script
tentative_shape <- 0.438167
tentative_scale <- 534.3507
tentative_kappa <- 0.1848126

hour_coefs_nat_df_0p <- harmonics_nat_df_0p %>%
  mutate(shape = tentative_shape + log_sl,
         scale = 1/((1/tentative_scale) - sl),
```

```
         kappa = tentative_kappa + cos_ta)

# save the coefficients to use in the simulations
write_csv(hour_coefs_nat_df_0p,
          paste0("ssf_coefficients/id", focal_id, "_0pDaily_coefs_", Sys.Date(), ".csv"))

# turning into a long data frame
hour_coefs_nat_long_0p <- pivot_longer(hour_coefs_nat_df_0p,
                                       cols = !1,
                                       names_to = "coef")
```

1p

```
hour_coefs_nat_df_1p <- harmonics_nat_df_1p %>%
  mutate(shape = tentative_shape + log_sl,
         scale = 1/((1/tentative_scale) - sl),
         kappa = tentative_kappa + cos_ta)

# save the coefficients to use in the simulations
write_csv(hour_coefs_nat_df_1p,
          paste0("ssf_coefficients/id", focal_id, "_1pDaily_coefs_",Sys.Date(), ".csv"))

# turning into a long data frame
hour_coefs_nat_long_1p <- pivot_longer(hour_coefs_nat_df_1p,
                                       cols = !1, names_to = "coef")
```

2p

```
hour_coefs_nat_df_2p <- harmonics_nat_df_2p %>%
  mutate(shape = tentative_shape + log_sl,
         scale = 1/((1/tentative_scale) - sl),
         kappa = tentative_kappa + cos_ta)

# save the coefficients to use in the simulations
write_csv(hour_coefs_nat_df_2p,
          paste0("ssf_coefficients/id", focal_id, "_2pDaily_coefs_",Sys.Date(), ".csv"))

# turning into a long data frame
hour_coefs_nat_long_2p <- pivot_longer(hour_coefs_nat_df_2p, cols = !1,
                                       names_to = "coef")
```

3p

```
hour_coefs_nat_df_3p <- harmonics_nat_df_3p %>%
  mutate(shape = tentative_shape + log_sl,
         scale = 1/((1/tentative_scale) - sl),
         kappa = tentative_kappa + cos_ta)

# save the coefficients to use in the simulations
write_csv(hour_coefs_nat_df_3p,
          paste0("ssf_coefficients/id", focal_id, "_3pDaily_coefs_", Sys.Date(), ".csv"))

# turning into a long data frame
hour_coefs_nat_long_3p <- pivot_longer(hour_coefs_nat_df_3p, cols = !1,
                                       names_to = "coef")
```

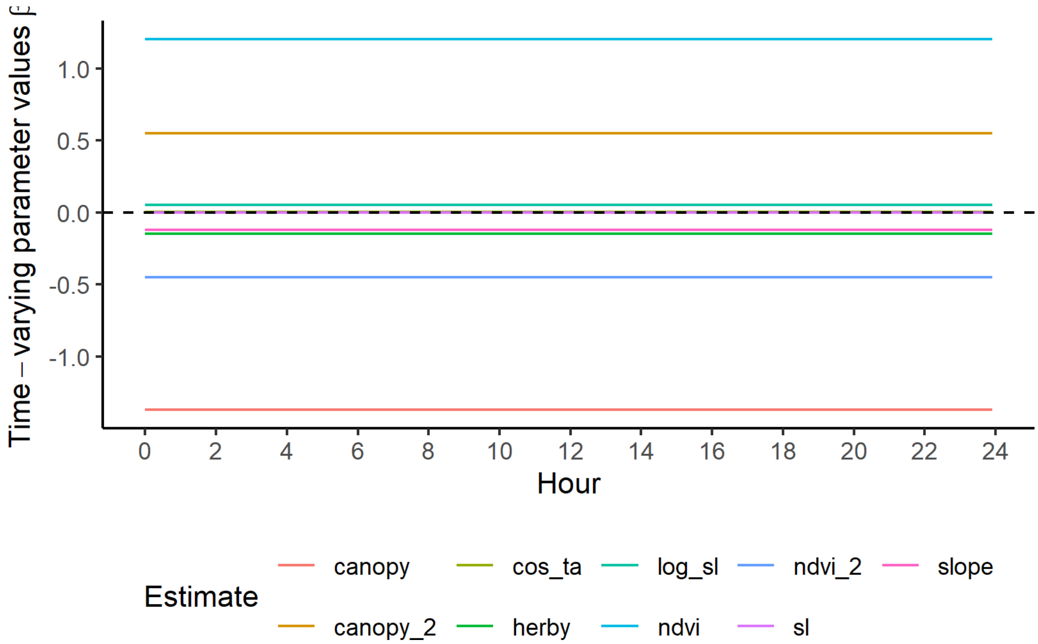
Plot the natural-scale temporally dynamic coefficients

Now that the coefficients are in their natural scales, they will be larger or smaller depending on the scale of the covariate.

Plot just the habitat selection coefficients.

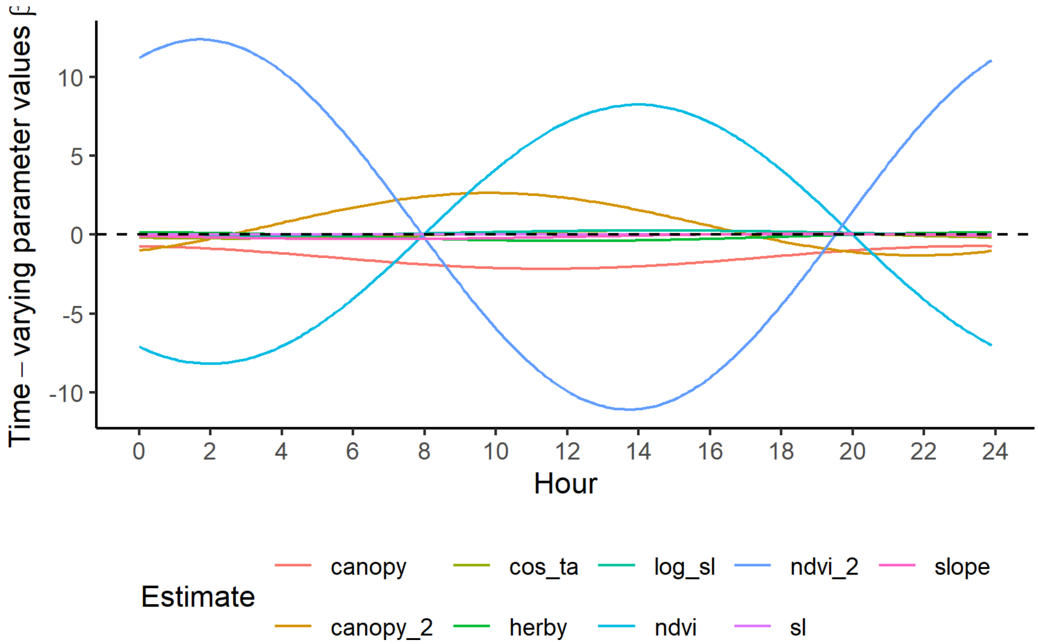
0p

```
ggplot() +
  geom_path(data = hour_coefs_nat_long_0p %>%
              filter(!coef %in% c("shape", "scale", "kappa")),
            aes(x = hour, y = value, colour = coef)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_color_discrete("Estimate") +
  theme_classic() +
  theme(legend.position = "bottom")
```



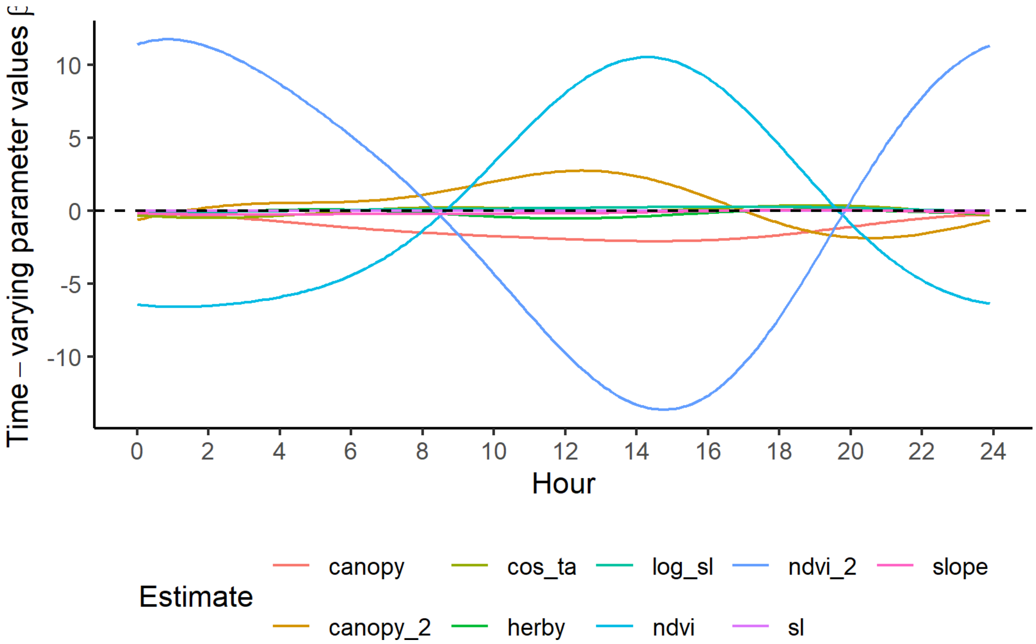
1p

```
ggplot() +
  geom_path(data = hour_coefs_nat_long_1p %>%
              filter(!coef %in% c("shape", "scale", "kappa")),
            aes(x = hour, y = value, colour = coef)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_color_discrete("Estimate") +
  theme_classic() +
  theme(legend.position = "bottom")
```



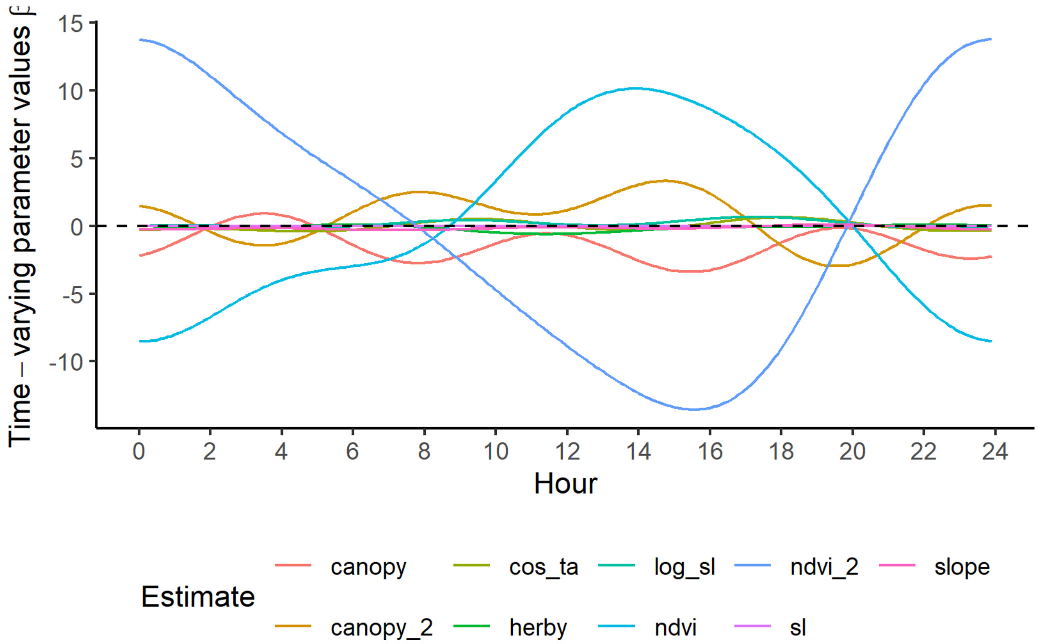
2p

```
ggplot() +
  geom_path(data = hour_coefs_nat_long_2p %>%
              filter(!coef %in% c("shape", "scale", "kappa")),
            aes(x = hour, y = value, colour = coef)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_color_discrete("Estimate") +
  theme_classic() +
  theme(legend.position = "bottom")
```



3p

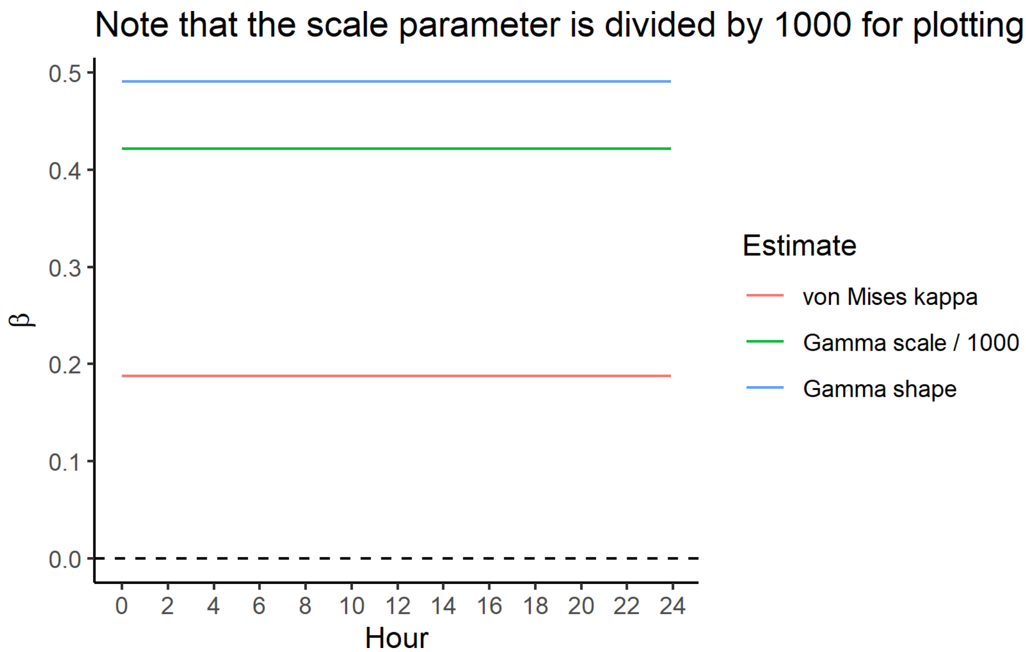
```r
ggplot() +
  geom_path(data = hour_coefs_nat_long_3p %>%
              filter(!coef %in% c("shape", "scale", "kappa")),
            aes(x = hour, y = value, colour = coef)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_color_discrete("Estimate") +
  theme_classic() +
  theme(legend.position = "bottom")
```



Plot only the temporally dynamic movement parameters

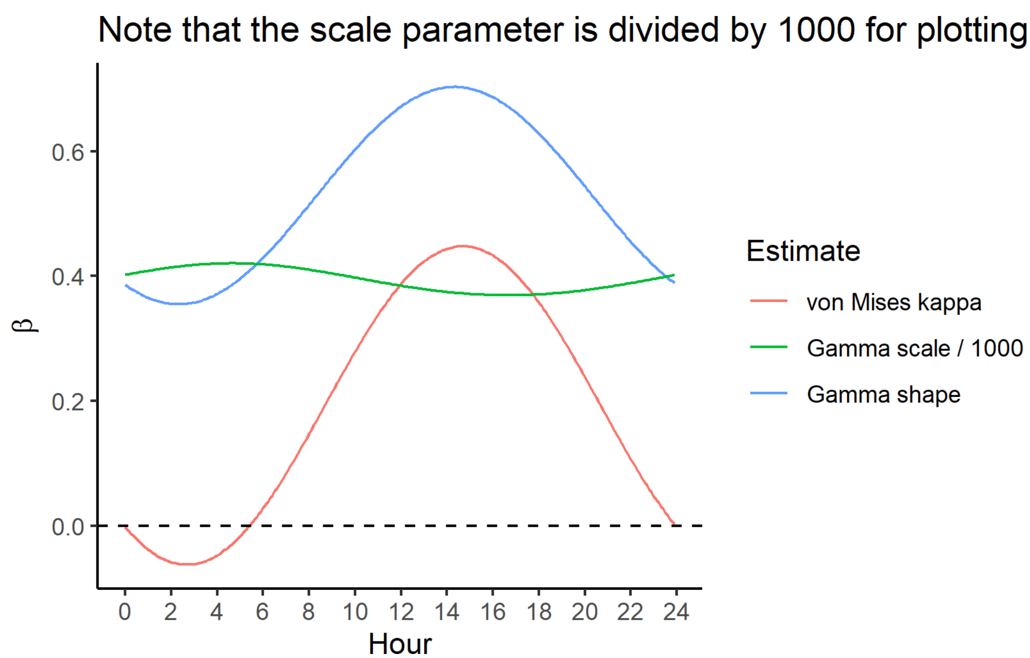
0p

```
ggplot() +
    geom_path(data = hour_coefs_nat_long_0p %>%
              filter(coef %in% c("shape", "kappa")),
              aes(x = hour, y = value, colour = coef)) +
  geom_path(data = hour_coefs_nat_long_0p %>%
              filter(coef == "scale"),
              aes(x = hour, y = value/1000, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  ggtitle("Note that the scale parameter is divided by 1000 for plotting") +
  scale_color_discrete("Estimate",
      labels = c("kappa" = "von Mises kappa",
                 "scale" = "Gamma scale / 1000",
                 "shape" = "Gamma shape")) +
    theme_classic() +
    theme(legend.position = "right")
```



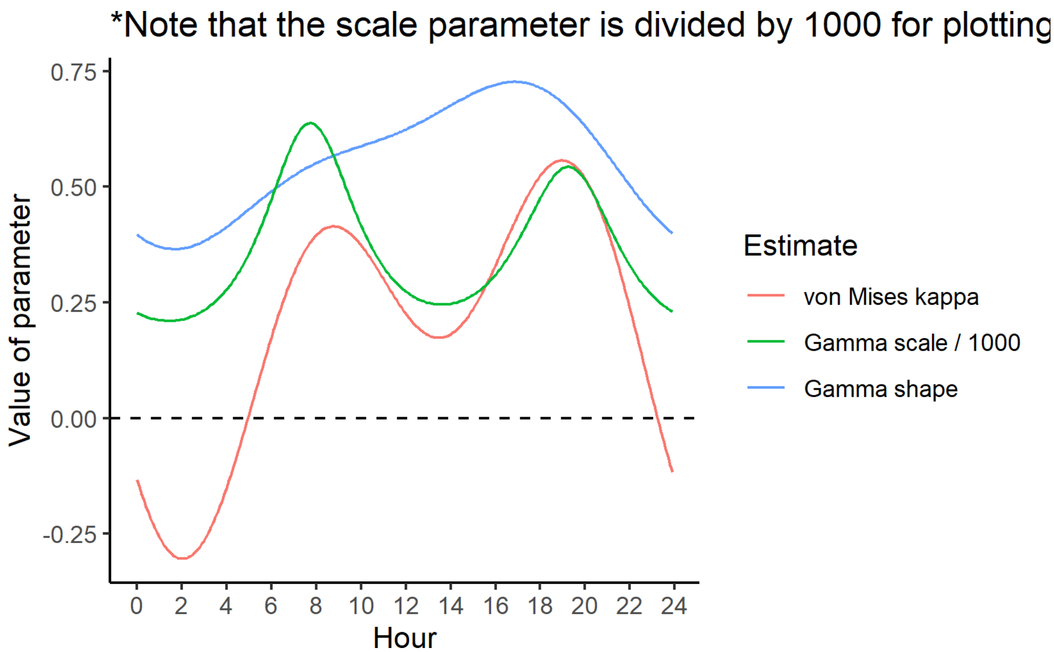
1p

```
ggplot() +
    geom_path(data = hour_coefs_nat_long_1p %>%
              filter(coef %in% c("shape", "kappa")),
              aes(x = hour, y = value, colour = coef)) +
  geom_path(data = hour_coefs_nat_long_1p %>%
              filter(coef == "scale"),
              aes(x = hour, y = value/1000, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  ggtitle("Note that the scale parameter is divided by 1000 for plotting") +
  scale_color_discrete("Estimate",
      labels = c("kappa" = "von Mises kappa",
                 "scale" = "Gamma scale / 1000",
                 "shape" = "Gamma shape")) +
    theme_classic() +
    theme(legend.position = "right")
```



2p

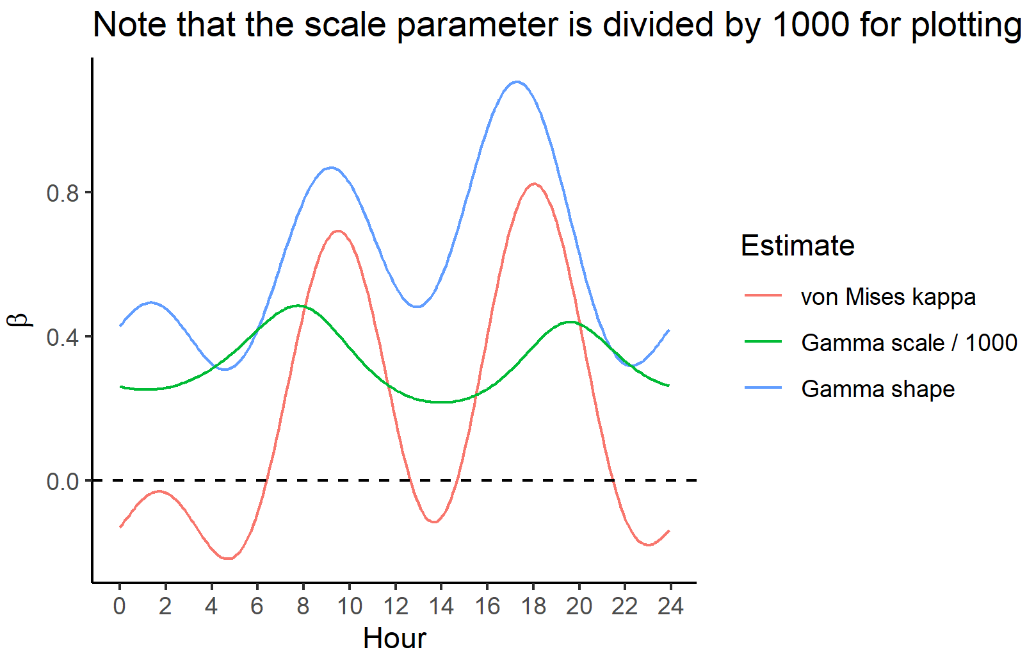
```
ggplot() +
    geom_path(data = hour_coefs_nat_long_2p %>%
              filter(coef %in% c("shape", "kappa")),
              aes(x = hour, y = value, colour = coef)) +
  geom_path(data = hour_coefs_nat_long_2p %>%
              filter(coef == "scale"),
              aes(x = hour, y = value/1000, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous("Value of parameter") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  ggtitle("*Note that the scale parameter is divided by 1000 for plotting") +
  scale_color_discrete("Estimate",
      labels = c("kappa" = "von Mises kappa",
                 "scale" = "Gamma scale / 1000",
                 "shape" = "Gamma shape")) +
    theme_classic() +
    theme(legend.position = "right")
```




```
# ggsave(paste0("outputs/plots/manuscript_figs_R2/temporal_mvmt_params_",
#           Sys.Date(), ".png"),
#   width=150, height=90, units="mm", dpi = 1000)
```

3p

```
ggplot() +
    geom_path(data = hour_coefs_nat_long_3p %>%
              filter(coef %in% c("shape", "kappa")),
              aes(x = hour, y = value, colour = coef)) +
  geom_path(data = hour_coefs_nat_long_3p %>%
              filter(coef == "scale"),
              aes(x = hour, y = value/1000, colour = coef)) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    scale_y_continuous(expression(beta)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  ggtitle("Note that the scale parameter is divided by 1000 for plotting") +
  scale_color_discrete("Estimate",
      labels = c("kappa" = "von Mises kappa",
                 "scale" = "Gamma scale / 1000",
                 "shape" = "Gamma shape")) +
    theme_classic() +
    theme(legend.position = "right")
```



Sample from temporally dynamic movement parameters

Here we sample from the movement kernel to generate a distribution of step lengths for each hour of the day, to assess how well it matches the observed step lengths. This is the 'selection-free' movement kernel, so the step lengths and turning angles from the simulations will be different, as the steps will be conditioned on the habitat, but this is a useful diagnostic to assess whether the harmonics are capturing the observed movement dynamics.

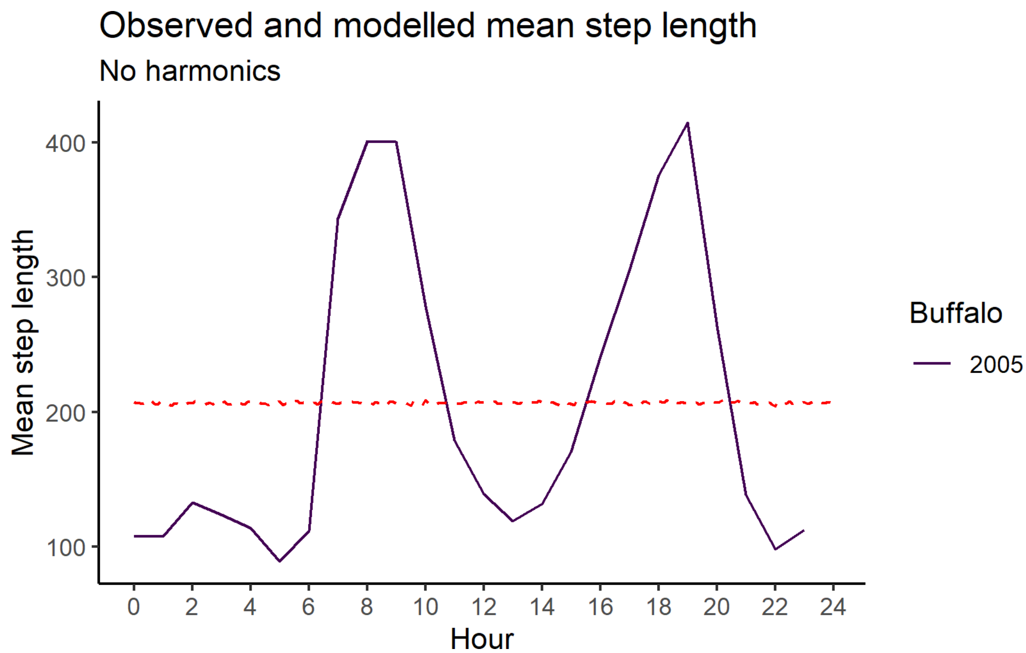
0p

```
# summarise the observed step lengths by hour
movement_summary_buffalo <- buffalo_data %>%
  filter(y == 1) %>%
  group_by(id, hour) %>%
  summarise(mean_sl = mean(sl), median_sl = median(sl))
```

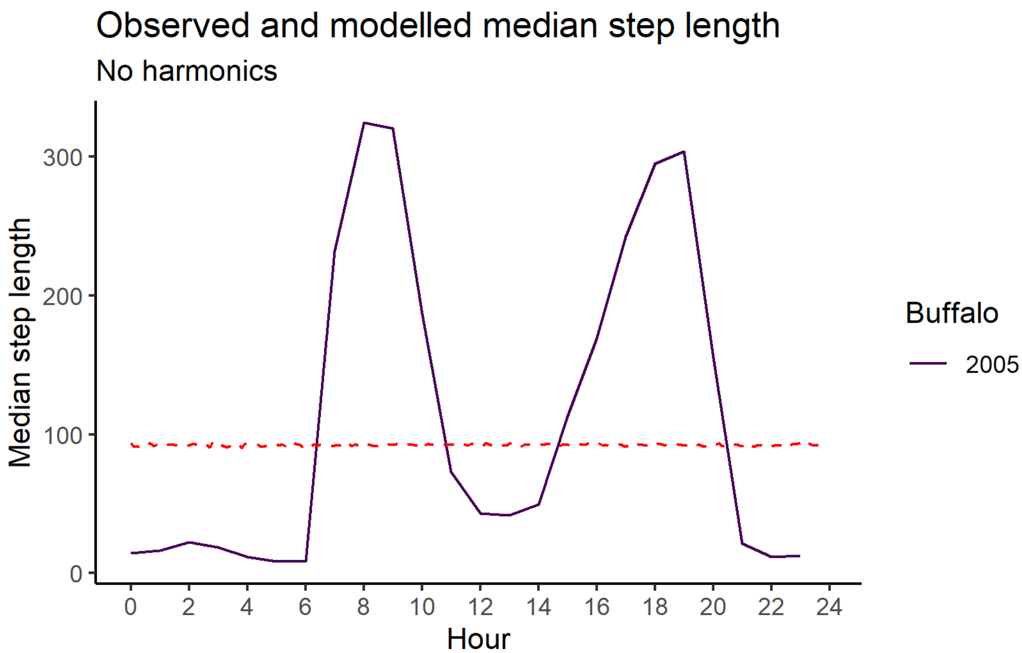
```
`summarise()` has grouped output by 'id'. You can override using the `.groups`
argument.
```

```
# number of samples at each hour (more = smoother plotting, but slower)
n <- 1e5

gamma_dist_list <- vector(mode = "list", length = nrow(hour_coefs_nat_df_0p))
```

```r
gamma_mean <- c()
gamma_median <- c()
gamma_ratio <- c()

for(hour_no in 1:nrow(hour_coefs_nat_df_0p)) {

  gamma_dist_list[[hour_no]] <- rgamma(n, shape = hour_coefs_nat_df_0p$shape[hour_no],
                                       scale = hour_coefs_nat_df_0p$scale[hour_no])

  gamma_mean[hour_no] <- mean(gamma_dist_list[[hour_no]])
  gamma_median[hour_no] <- median(gamma_dist_list[[hour_no]])
  gamma_ratio[hour_no] <- gamma_mean[hour_no] / gamma_median[hour_no]

}

gamma_df_0p <- data.frame(model = "0p",
                          hour = hour_coefs_nat_df_0p$hour,
                          mean = gamma_mean,
                          median = gamma_median,
                          ratio = gamma_ratio)

mean_sl_0p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = mean_sl, colour = factor(id))) +
  geom_path(data = gamma_df_0p,
            aes(x = hour, y = mean), colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Mean step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled mean step length",
          subtitle = "No harmonics") +
  theme_classic() +
  theme(legend.position = "right")

mean_sl_0p
```




```
median_sl_0p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = median_sl, colour = factor(id))) +
  geom_path(data = gamma_df_0p, aes(x = hour, y = median),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Median step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled median step length",
          subtitle = "No harmonics") +
  theme_classic() +
  theme(legend.position = "right")

median_sl_0p
```




```
# comparing the mean and median step lengths across all hours
# across the hours by individual buffalo
buffalo_data_all %>% filter(y == 1) %>%  group_by(id) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 14 x 4
      id mean_sl median_sl ratio
   <dbl>   <dbl>     <dbl> <dbl>
 1  2005    205.      89.7  2.29
 2  2014    135.      13.5 10.0
 3  2018    252.     103.   2.44
 4  2021    183.      94.8  1.93
 5  2022    219.      79.8  2.74
 6  2024    211.      70.9  2.97
 7  2039    357.     124.   2.87
 8  2154    189.      88.9  2.13
 9  2158    219.      82.1  2.67
10  2223    249.      80.2  3.10
11  2327    199.      46.0  4.32
12  2354    232.      79.7  2.91
13  2387    328.     108.   3.03
14  2393    322.     127.   2.53
```

```
# all buffalo
buffalo_data_all %>% filter(y == 1) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 1 x 3
  mean_sl median_sl ratio
    <dbl>     <dbl> <dbl>
1    234.      82.3  2.84
```

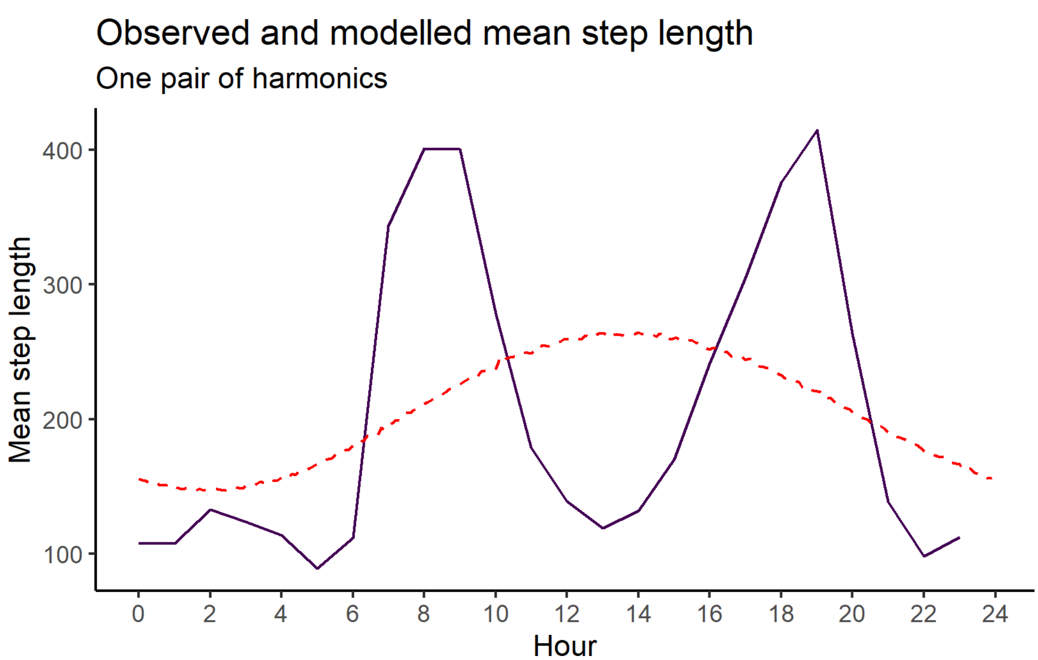
```
# fitted model
gamma_df_0p %>% summarise(mean_mean = mean(mean),
                         median_mean = mean(median),
                         ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1  206.6799    92.33112   2.238464
```

1p

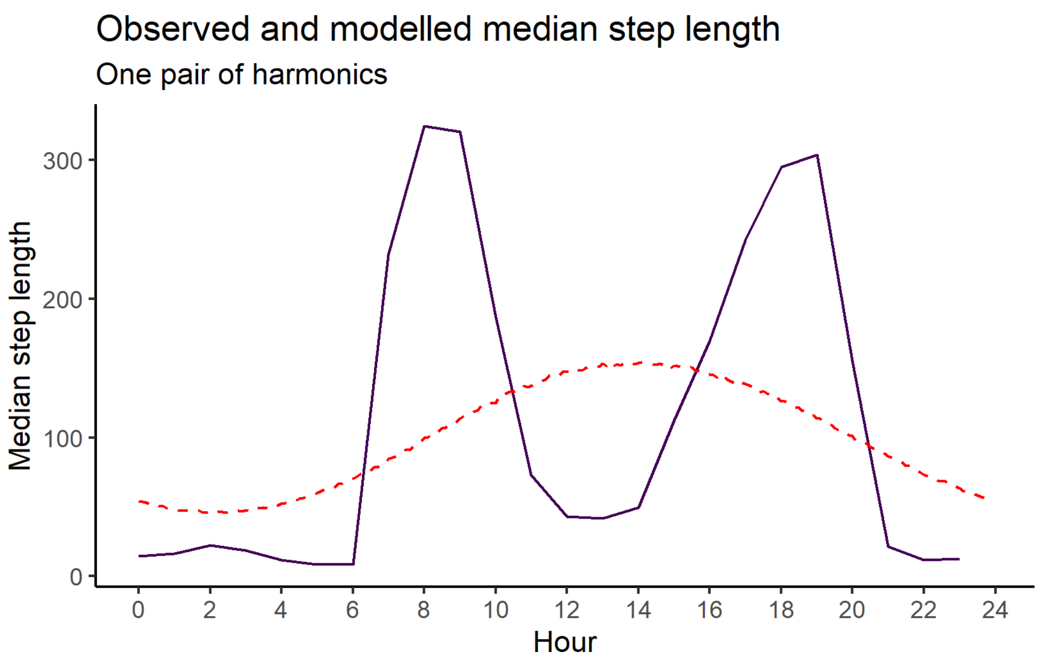
```
gamma_dist_list <- vector(mode = "list", length = nrow(hour_coefs_nat_df_1p))
gamma_mean <- c()
gamma_median <- c()
gamma_ratio <- c()

for(hour_no in 1:nrow(hour_coefs_nat_df_1p)) {

  gamma_dist_list[[hour_no]] <- rgamma(n,
                                       shape = hour_coefs_nat_df_1p$shape[hour_no],
                                       scale = hour_coefs_nat_df_1p$scale[hour_no])

  gamma_mean[hour_no] <- mean(gamma_dist_list[[hour_no]])
  gamma_median[hour_no] <- median(gamma_dist_list[[hour_no]])
  gamma_ratio[hour_no] <- gamma_mean[hour_no] / gamma_median[hour_no]

}

gamma_df_1p <- data.frame(model = "1p",
                          hour = hour_coefs_nat_df_1p$hour,
                          mean = gamma_mean,
                          median = gamma_median,
```

```
                          ratio = gamma_ratio)

mean_sl_1p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = mean_sl, colour = factor(id))) +
  geom_path(data = gamma_df_1p,
            aes(x = hour, y = mean),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Mean step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled mean step length",
          subtitle = "One pair of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

mean_sl_1p
```




```
median_sl_1p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = median_sl, colour = factor(id))) +
  geom_path(data = gamma_df_1p,
            aes(x = hour, y = median),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
```

```
  scale_y_continuous("Median step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled median step length",
          subtitle = "One pair of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

median_sl_1p
```




```
# across the hours
buffalo_data_all %>% filter(y == 1) %>%  group_by(id) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 14 x 4
      id mean_sl median_sl ratio
   <dbl>   <dbl>     <dbl> <dbl>
 1  2005    205.      89.7  2.29
 2  2014    135.      13.5 10.0
 3  2018    252.     103.   2.44
 4  2021    183.      94.8  1.93
 5  2022    219.      79.8  2.74
 6  2024    211.      70.9  2.97
 7  2039    357.     124.   2.87
```

```
 8  2154    189.      88.9  2.13
 9  2158    219.      82.1  2.67
10  2223    249.      80.2  3.10
11  2327    199.      46.0  4.32
12  2354    232.      79.7  2.91
13  2387    328.     108.   3.03
14  2393    322.     127.   2.53
```

```
buffalo_data_all %>% filter(y == 1) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 1 x 3
  mean_sl median_sl ratio
    <dbl>     <dbl> <dbl>
1    234.      82.3  2.84
```

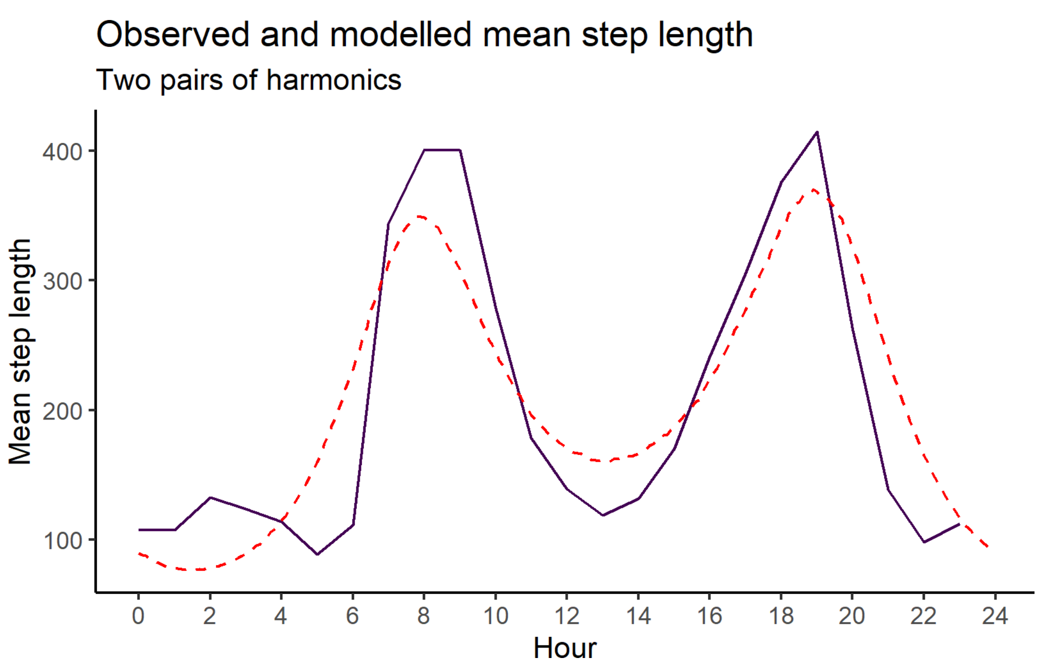
```
gamma_df_1p %>% summarise(mean_mean = mean(mean),
                          median_mean = mean(median),
                          ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1  206.7879    99.65689   2.074999
```

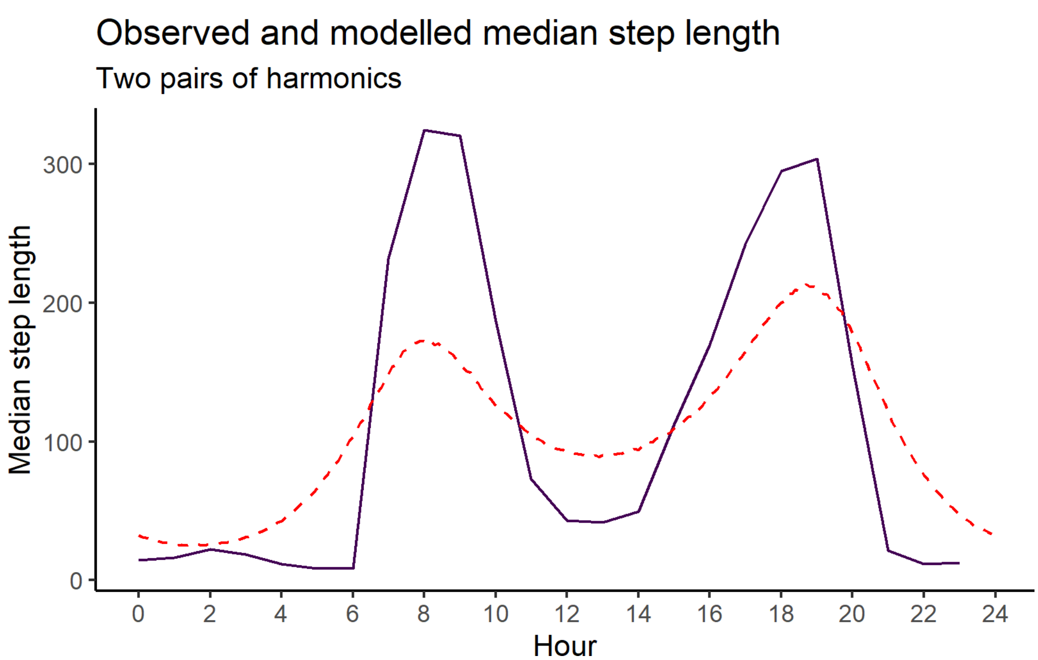
2p

```
gamma_dist_list <- vector(mode = "list", length = nrow(hour_coefs_nat_df_2p))
gamma_mean <- c()
gamma_median <- c()
gamma_ratio <- c()

for(hour_no in 1:nrow(hour_coefs_nat_df_2p)) {

gamma_dist_list[[hour_no]] <- rgamma(n,
                                     shape = hour_coefs_nat_df_2p$shape[hour_no],
                                     scale = hour_coefs_nat_df_2p$scale[hour_no])

gamma_mean[hour_no] <- mean(gamma_dist_list[[hour_no]])
gamma_median[hour_no] <- median(gamma_dist_list[[hour_no]])
gamma_ratio[hour_no] <- gamma_mean[hour_no] / gamma_median[hour_no]
```

```
}

gamma_df_2p <- data.frame(model = "2p",
                          hour = hour_coefs_nat_df_2p$hour,
                          mean = gamma_mean,
                          median = gamma_median,
                          ratio = gamma_ratio)

mean_sl_2p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = mean_sl, colour = factor(id))) +
  geom_path(data = gamma_df_2p,
            aes(x = hour, y = mean),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Mean step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled mean step length",
          subtitle = "Two pairs of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

mean_sl_2p
```



```r
median_sl_2p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = median_sl, colour = factor(id))) +
  geom_path(data = gamma_df_2p,
            aes(x = hour, y = median),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Median step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled median step length",
          subtitle = "Two pairs of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

median_sl_2p
```




```r
# across the hours
buffalo_data_all %>% filter(y == 1) %>%  group_by(id) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 14 x 4
      id mean_sl median_sl ratio
   <dbl>   <dbl>     <dbl> <dbl>
```

```
 1  2005    205.      89.7  2.29
 2  2014    135.      13.5 10.0
 3  2018    252.     103.   2.44
 4  2021    183.      94.8  1.93
 5  2022    219.      79.8  2.74
 6  2024    211.      70.9  2.97
 7  2039    357.     124.   2.87
 8  2154    189.      88.9  2.13
 9  2158    219.      82.1  2.67
10  2223    249.      80.2  3.10
11  2327    199.      46.0  4.32
12  2354    232.      79.7  2.91
13  2387    328.     108.   3.03
14  2393    322.     127.   2.53
```

```
buffalo_data_all %>% filter(y == 1) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 1 x 3
  mean_sl median_sl ratio
    <dbl>     <dbl> <dbl>
1    234.      82.3  2.84
```

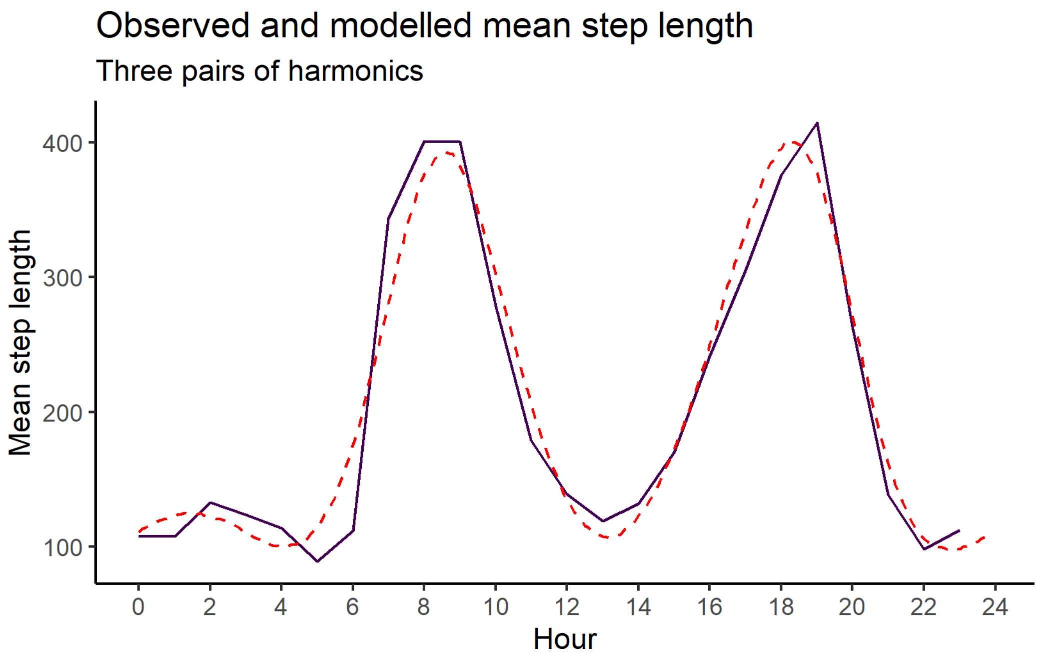
```
gamma_df_2p %>% summarise(mean_mean = mean(mean),
                          median_mean = mean(median),
                          ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1   208.324    106.3893    1.95813
```

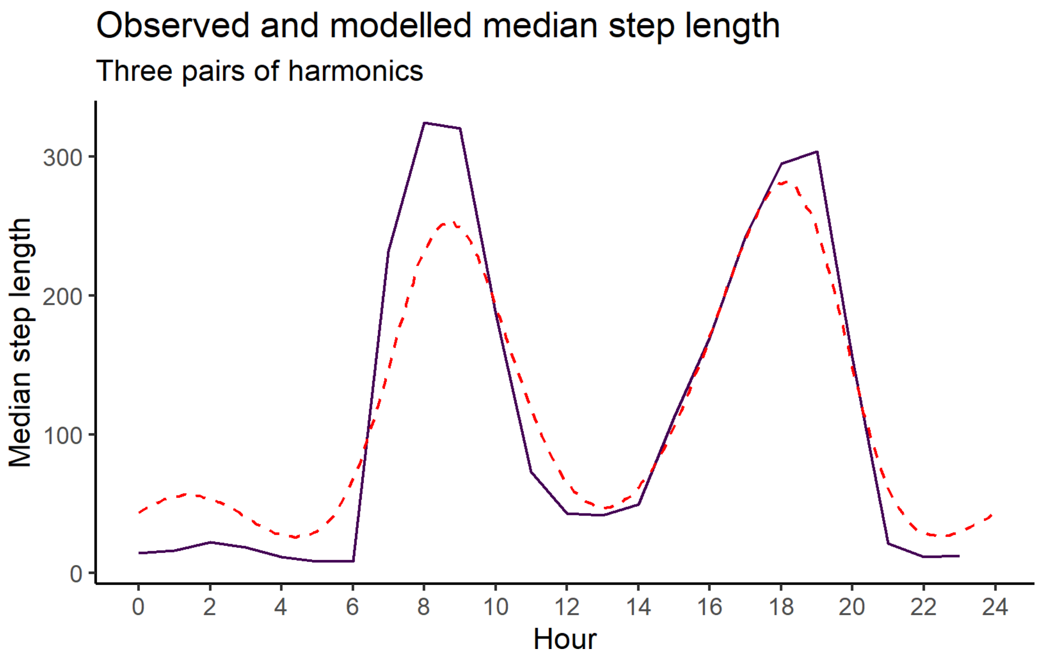
3p

```
gamma_dist_list <- vector(mode = "list", length = nrow(hour_coefs_nat_df_3p))
gamma_mean <- c()
gamma_median <- c()
gamma_ratio <- c()

for(hour_no in 1:nrow(hour_coefs_nat_df_3p)) {

gamma_dist_list[[hour_no]] <- rgamma(n,
```

```
                                        shape = hour_coefs_nat_df_3p$shape[hour_no],
                                        scale = hour_coefs_nat_df_3p$scale[hour_no])

gamma_mean[hour_no] <- mean(gamma_dist_list[[hour_no]])
gamma_median[hour_no] <- median(gamma_dist_list[[hour_no]])
gamma_ratio[hour_no] <- gamma_mean[hour_no] / gamma_median[hour_no]

}

gamma_df_3p <- data.frame(model = "3p",
                          hour = hour_coefs_nat_df_3p$hour,
                          mean = gamma_mean,
                          median = gamma_median,
                          ratio = gamma_ratio)

mean_sl_3p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = mean_sl, colour = factor(id))) +
  geom_path(data = gamma_df_3p,
            aes(x = hour, y = mean),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Mean step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled mean step length",
          subtitle = "Three pairs of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

mean_sl_3p
```




```
median_sl_3p <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = median_sl, colour = factor(id))) +
  geom_path(data = gamma_df_3p,
            aes(x = hour, y = median),
            colour = "red", linetype = "dashed") +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Median step length") +
  scale_colour_viridis_d("Buffalo") +
  ggtitle("Observed and modelled median step length",
          subtitle = "Three pairs of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

median_sl_3p
```




```
# across the hours
buffalo_data_all %>% filter(y == 1) %>%  group_by(id) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 14 x 4
      id mean_sl median_sl ratio
   <dbl>   <dbl>     <dbl> <dbl>
 1  2005    205.      89.7  2.29
 2  2014    135.      13.5 10.0
 3  2018    252.     103.   2.44
 4  2021    183.      94.8  1.93
 5  2022    219.      79.8  2.74
 6  2024    211.      70.9  2.97
 7  2039    357.     124.   2.87
 8  2154    189.      88.9  2.13
 9  2158    219.      82.1  2.67
10  2223    249.      80.2  3.10
11  2327    199.      46.0  4.32
12  2354    232.      79.7  2.91
13  2387    328.     108.   3.03
14  2393    322.     127.   2.53
```

```r
buffalo_data_all %>% filter(y == 1) %>%
  summarise(mean_sl = mean(sl),
            median_sl = median(sl),
            ratio = mean_sl/median_sl)
```

```
# A tibble: 1 x 3
  mean_sl median_sl ratio
    <dbl>     <dbl> <dbl>
1    234.      82.3  2.84
```

```r
gamma_df_3p %>% summarise(mean_mean = mean(mean),
                          median_mean = mean(median),
                          ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1  205.7595    114.3458    1.79945
```

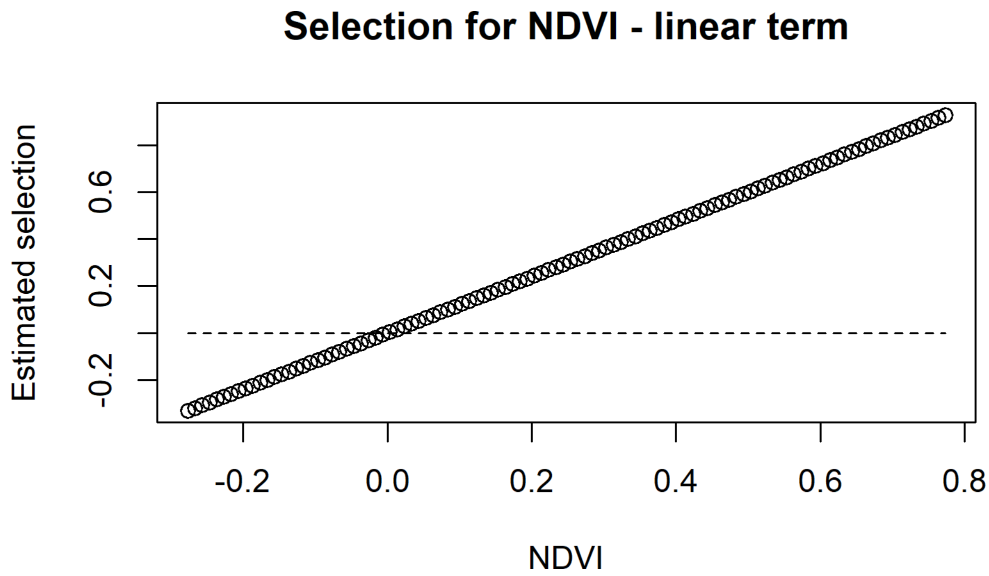
Creating selection surfaces

As we have both quadratic and harmonic terms in the model, we can reconstruct a 'selection surface' to visualise how the animal's respond to environmental features changes through time.

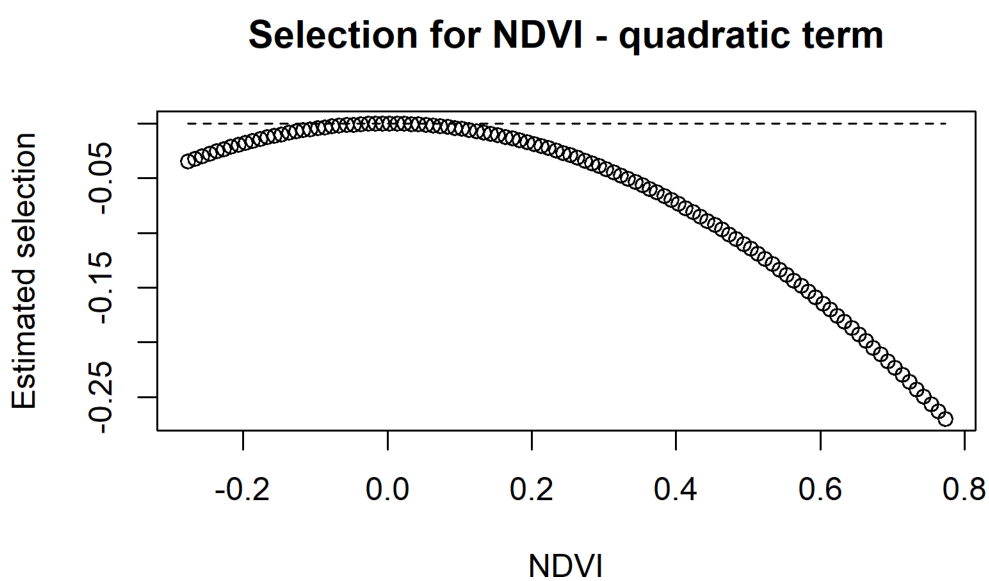
To illustrate, if we don't have temporal dynamics (as is the case for this model), then we have a coefficient for the linear term and a coefficient for the quadratic term. Using these, we can plot the selection curve at the scale of the environmental variable (in this case NDVI).

Using the natural scale coefficients from the model:

```r
# first get a sequence of NDVI values,
# starting from the minimum observed in the data to the maximum
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# take the coefficients from the model and calculation the selection value
# for every NDVI value in this sequence

# we can separate to the linear term
ndvi_linear_selection <- hour_coefs_nat_df_0p$ndvi[1] * ndvi_seq
plot(x = ndvi_seq, y = ndvi_linear_selection,
     main = "Selection for NDVI - linear term",
     xlab = "NDVI", ylab = "Estimated selection")
lines(ndvi_seq, rep(0,length(ndvi_seq)), lty = "dashed")
```

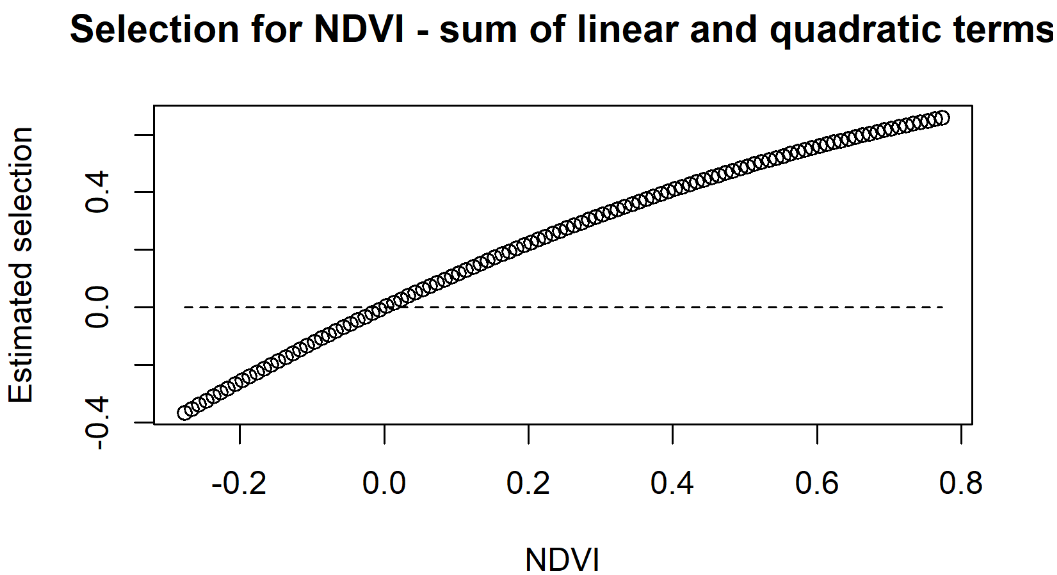



```
# and the quadratic term
ndvi_quadratic_selection <- (hour_coefs_nat_df_0p$ndvi_2[1] * (ndvi_seq ^ 2))
plot(x = ndvi_seq, y = ndvi_quadratic_selection,
     main = "Selection for NDVI - quadratic term",
     xlab = "NDVI", ylab = "Estimated selection")
lines(ndvi_seq, rep(0,length(ndvi_seq)), lty = "dashed")
```




```
# and the sum of both
ndvi_sum_selection <- ndvi_linear_selection + ndvi_quadratic_selection
plot(x = ndvi_seq, y = ndvi_sum_selection,
     main = "Selection for NDVI - sum of linear and quadratic terms",
```

```
     xlab = "NDVI", ylab = "Estimated selection")
lines(ndvi_seq, rep(0,length(ndvi_seq)), lty = "dashed")
```



When there are no temporal dynamics, then this quadratic curve will be the same throughout the day, but when we have temporally dynamic coefficients for both the linear term and the quadratic term, then we will have a curves that vary continuously throughout the day, which we can visualise as a selection surface.

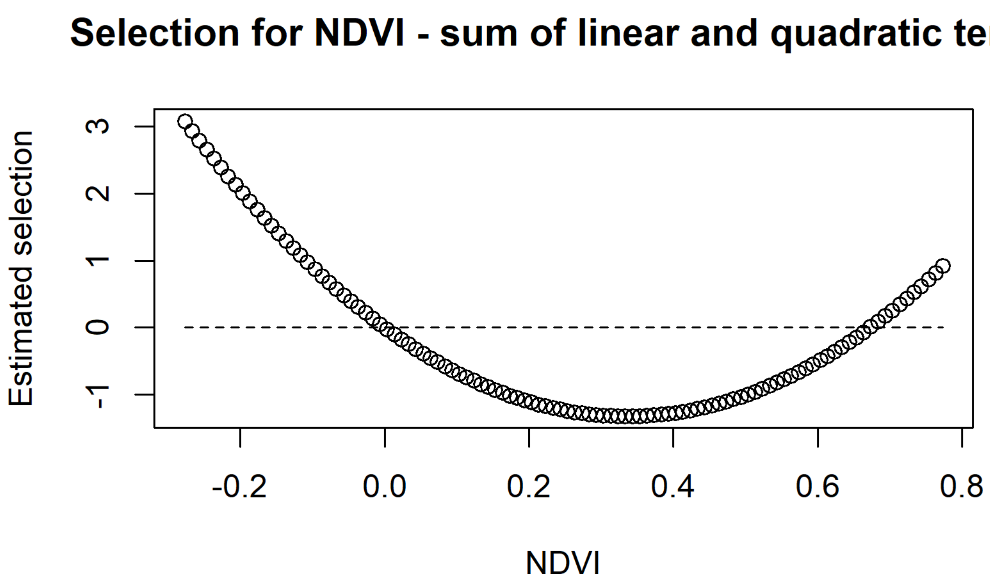
Here we illustrate for the model with 2 pairs of harmonic terms.

For brevity we won't plot the linear and quadratic terms separately, but we can do so if needed.

First for **Hour 3**

```
hour_no <- 3

# we can separate to the linear term
ndvi_linear_selection <-
  hour_coefs_nat_df_1p$ndvi[which(hour_coefs_nat_df_1p$hour == hour_no)] * ndvi_seq
# plot(x = ndvi_seq, y = ndvi_linear_selection,
#      main = "Selection for NDVI - linear term",
#      xlab = "NDVI", ylab = "Estimated selection")

# and the quadratic term
ndvi_quadratic_selection <-
  (hour_coefs_nat_df_1p$ndvi_2[which(hour_coefs_nat_df_1p$hour == hour_no)] * (ndvi_seq ^ 2)
# plot(x = ndvi_seq, y = ndvi_quadratic_selection,
#      main = "Selection for NDVI - quadratic term",
```

```
#      xlab = "NDVI", ylab = "Estimated selection")

# and the sum of both
ndvi_sum_selection <- ndvi_linear_selection + ndvi_quadratic_selection
plot(x = ndvi_seq, y = ndvi_sum_selection,
     main = "Selection for NDVI - sum of linear and quadratic terms",
     xlab = "NDVI", ylab = "Estimated selection")
lines(ndvi_seq, rep(0,length(ndvi_seq)), lty = "dashed")
```

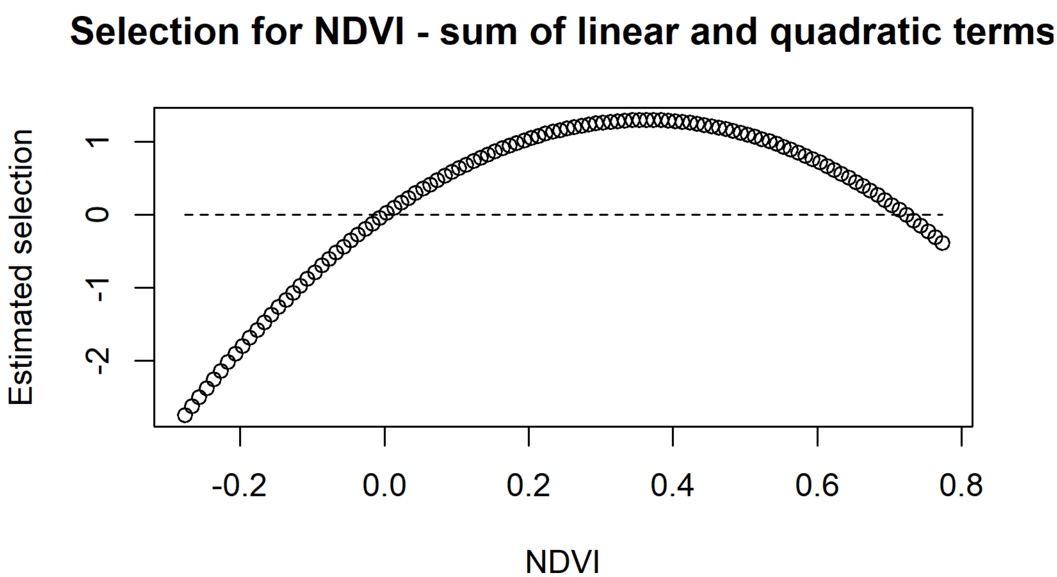


We can see that the coefficient at hour 3 shows highest selection for NDVI values slightly above 0.2, and the coefficient is mostly negative.

Secondly for **Hour 12**

```
hour_no <- 12

# we can separate to the linear term
ndvi_linear_selection <-
  hour_coefs_nat_df_1p$ndvi[which(hour_coefs_nat_df_1p$hour == hour_no)] * ndvi_seq
# plot(x = ndvi_seq, y = ndvi_linear_selection,
#      main = "Selection for NDVI - linear term",
#      xlab = "NDVI", ylab = "Estimated selection")

# and the quadratic term
ndvi_quadratic_selection <-
  (hour_coefs_nat_df_1p$ndvi_2[which(hour_coefs_nat_df_1p$hour == hour_no)] * (ndvi_seq ^ 2)
# plot(x = ndvi_seq, y = ndvi_quadratic_selection,
#      main = "Selection for NDVI - quadratic term",
#      xlab = "NDVI", ylab = "Estimated selection")
```

```
# and the sum of both
ndvi_sum_selection <- ndvi_linear_selection + ndvi_quadratic_selection
plot(x = ndvi_seq, y = ndvi_sum_selection,
     main = "Selection for NDVI - sum of linear and quadratic terms",
     xlab = "NDVI", ylab = "Estimated selection")
lines(ndvi_seq, rep(0,length(ndvi_seq)), lty = "dashed")
```



Whereas for hour 12, the coefficient shows highest selection for NDVI values slightly above 0.4, and the coefficient is positive for NDVI values above 0.

We can imagine viewing these plots for every hour of the day, where each hour has a different quadratic curve, but this would be a lot of plots. We can also see it as a 3D surface, where the x-axis is the hour of the day, the y-axis is the NDVI value, and the z-axis (colour) is the coefficient value.

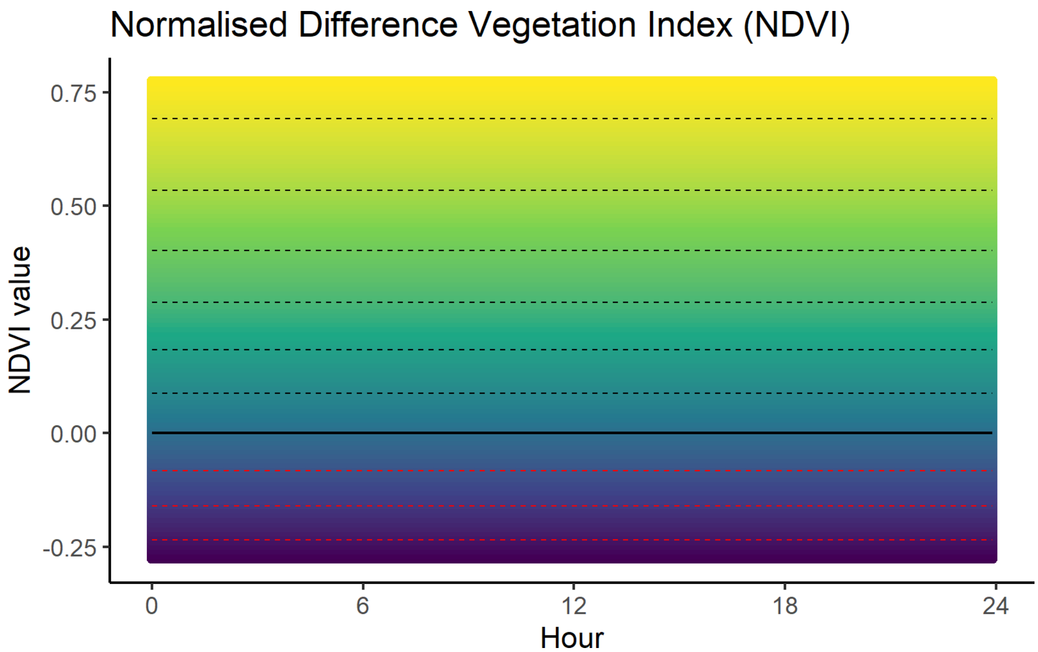
We simply index over the linear and quadratic terms and calculate the coefficient values at every time point.

NDVI selection surface

0p

```
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# Create empty data frame
```

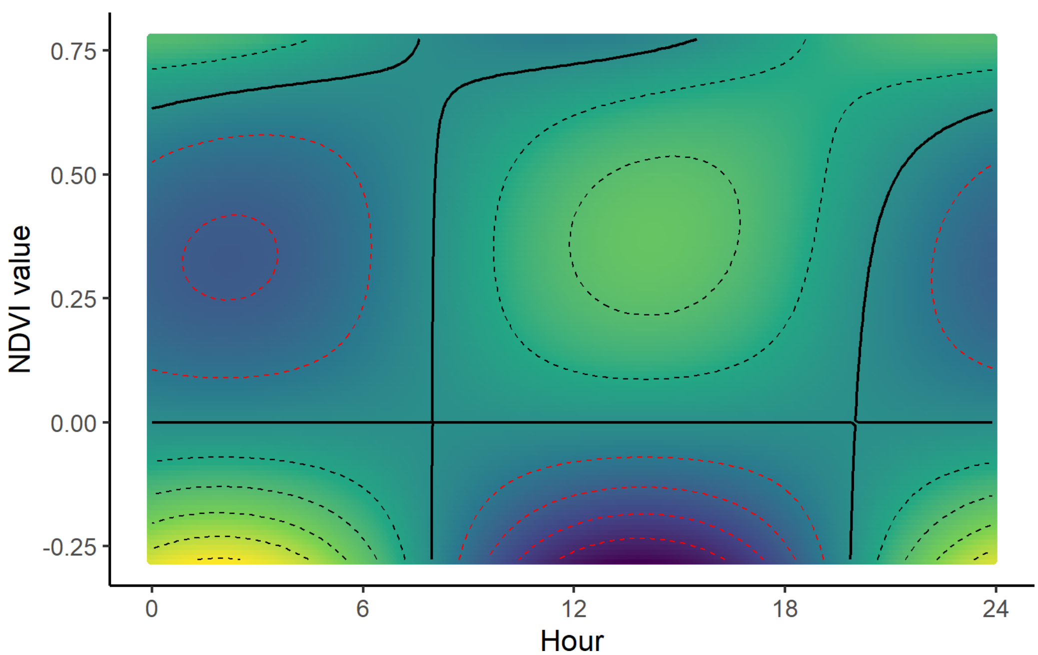
```
ndvi_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_0p),
                                       nrow = length(ndvi_seq)))

# loop over each time increment, calculating the selection values for each NDVI value
# and storing each time increment as a column in a dataframe that we can use for plotting
for(i in 1:nrow(hour_coefs_nat_df_0p)) {
  # Assign the vector as a column to the dataframe
  ndvi_fresponse_df[,i] <- (hour_coefs_nat_df_0p$ndvi[i] * ndvi_seq) +
    (hour_coefs_nat_df_0p$ndvi_2[i] * (ndvi_seq ^ 2))
}

ndvi_fresponse_df <- data.frame(ndvi_seq, ndvi_fresponse_df)
colnames(ndvi_fresponse_df) <- c("ndvi", hour)
ndvi_fresponse_long <- pivot_longer(ndvi_fresponse_df,
                                    cols = !1, names_to = "hour")

ndvi_contour_max <- max(ndvi_fresponse_long$value) # 0.7890195
ndvi_contour_min <- min(ndvi_fresponse_long$value) # -0.7945691
ndvi_contour_increment <- (ndvi_contour_max-ndvi_contour_min)/10

ndvi_quad_0p <- ggplot(data = ndvi_fresponse_long,
                       aes(x = as.numeric(hour), y = ndvi)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(ndvi_contour_increment,
                            ndvi_contour_max,
                            ndvi_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-ndvi_contour_increment,
                            ndvi_contour_min,
                            -ndvi_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("NDVI value", breaks = seq(-1, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  ggtitle("Normalised Difference Vegetation Index (NDVI)") +
  theme_classic() +
  theme(legend.position = "none")

ndvi_quad_0p
```



1p

```
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# Create empty data frame
ndvi_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_1p),
                                       nrow = length(ndvi_seq)))

for(i in 1:nrow(hour_coefs_nat_df_1p)) {
  # Assign the vector as a column to the dataframe
  ndvi_fresponse_df[,i] <- (hour_coefs_nat_df_1p$ndvi[i] * ndvi_seq) +
    (hour_coefs_nat_df_1p$ndvi_2[i] * (ndvi_seq ^ 2))
}

ndvi_fresponse_df <- data.frame(ndvi_seq, ndvi_fresponse_df)
colnames(ndvi_fresponse_df) <- c("ndvi", hour)
ndvi_fresponse_long <- pivot_longer(ndvi_fresponse_df, cols = !1, names_to = "hour")

ndvi_contour_max <- max(ndvi_fresponse_long$value) # 0.7890195
ndvi_contour_min <- min(ndvi_fresponse_long$value) # -0.7945691
ndvi_contour_increment <- (ndvi_contour_max-ndvi_contour_min)/10

ndvi_quad_1p <- ggplot(data = ndvi_fresponse_long,
```

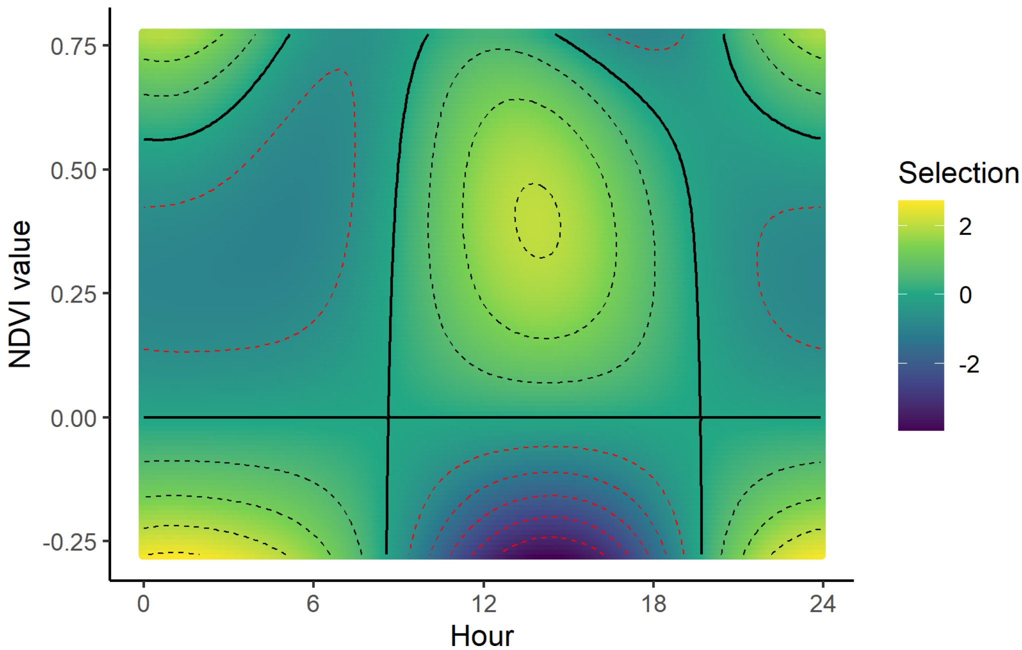
```
                          aes(x = as.numeric(hour), y = ndvi)) +
  geom_point(aes(colour = value)) + # colour = "white"
  geom_contour(aes(z = value),
               breaks = seq(ndvi_contour_increment,
                            ndvi_contour_max,
                            ndvi_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-ndvi_contour_increment,
                            ndvi_contour_min,
                            -ndvi_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("NDVI value", breaks = seq(-1, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Normalised Difference Vegetation Index (NDVI)") +
  theme_classic() +
  theme(legend.position = "none")

ndvi_quad_1p
```



2p

```
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# Create empty data frame
ndvi_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_2p),
                                       nrow = length(ndvi_seq)))

for(i in 1:nrow(hour_coefs_nat_df_2p)) {
  # Assign the vector as a column to the dataframe
  ndvi_fresponse_df[,i] <- (hour_coefs_nat_df_2p$ndvi[i] * ndvi_seq) +
    (hour_coefs_nat_df_2p$ndvi_2[i] * (ndvi_seq ^ 2))
}

ndvi_fresponse_df <- data.frame(ndvi_seq, ndvi_fresponse_df)
colnames(ndvi_fresponse_df) <- c("ndvi", hour)
ndvi_fresponse_long <- pivot_longer(ndvi_fresponse_df, cols = !1,
                                    names_to = "hour")

ndvi_contour_max <- max(ndvi_fresponse_long$value) # 0.7890195
ndvi_contour_min <- min(ndvi_fresponse_long$value) # -0.7945691
ndvi_contour_increment <- (ndvi_contour_max-ndvi_contour_min)/10

ndvi_quad_2p <- ggplot(data = ndvi_fresponse_long,
                       aes(x = as.numeric(hour), y = ndvi)) +
  geom_point(aes(colour = value)) + # colour = "white"
  geom_contour(aes(z = value),
               breaks = seq(ndvi_contour_increment,
                            ndvi_contour_max,
                            ndvi_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-ndvi_contour_increment,
                            ndvi_contour_min,
                            -ndvi_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("NDVI value", breaks = seq(-1, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Normalised Difference Vegetation Index (NDVI)") +
  theme_classic() +
```

```
  theme(legend.position = "right")

ndvi_quad_2p
```

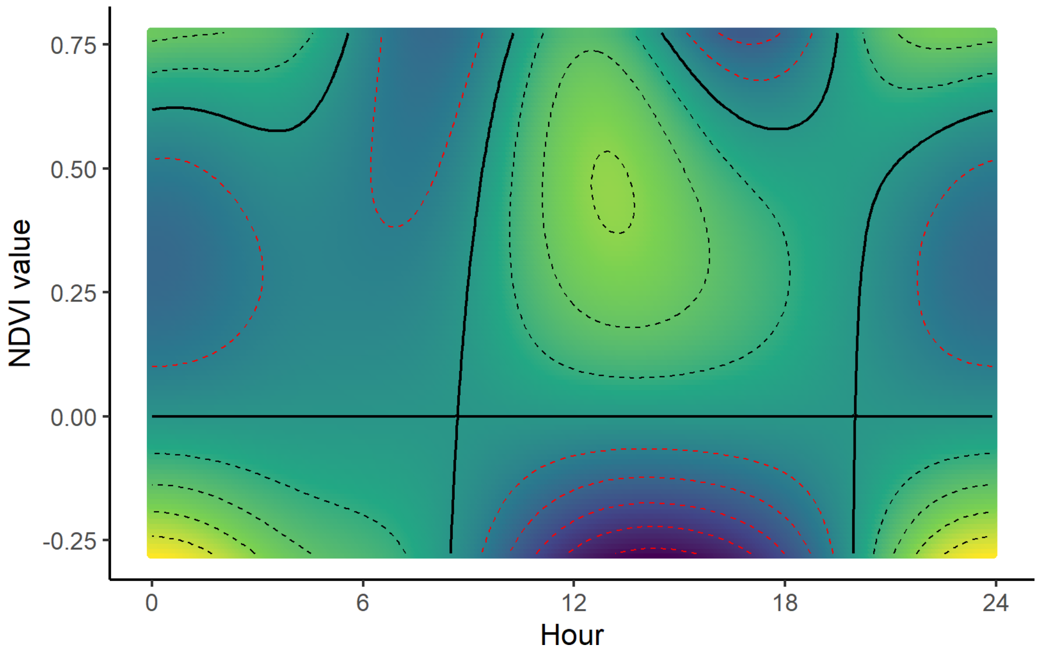



```
# ggsave(paste0("outputs/plots/manuscript_figs_R2/ndvi_selection_surface_legend_",
#          Sys.Date(), ".png"),
#   width=170, height=90, units="mm", dpi = 1000)
```

3p

```
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# Create empty data frame
ndvi_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_3p),
                                       nrow = length(ndvi_seq)))

for(i in 1:nrow(hour_coefs_nat_df_3p)) {
  # Assign the vector as a column to the dataframe
  ndvi_fresponse_df[,i] <- (hour_coefs_nat_df_3p$ndvi[i] * ndvi_seq) +
    (hour_coefs_nat_df_3p$ndvi_2[i] * (ndvi_seq ^ 2))
}
```

```r
ndvi_fresponse_df <- data.frame(ndvi_seq, ndvi_fresponse_df)
colnames(ndvi_fresponse_df) <- c("ndvi", hour)
ndvi_fresponse_long <- pivot_longer(ndvi_fresponse_df, cols = !1,
                                    names_to = "hour")

ndvi_contour_max <- max(ndvi_fresponse_long$value) # 0.7890195
ndvi_contour_min <- min(ndvi_fresponse_long$value) # -0.7945691
ndvi_contour_increment <- (ndvi_contour_max-ndvi_contour_min)/10

ndvi_quad_3p <- ggplot(data = ndvi_fresponse_long,
                       aes(x = as.numeric(hour), y = ndvi)) +
  geom_point(aes(colour = value)) + # colour = "white"
  geom_contour(aes(z = value),
               breaks = seq(ndvi_contour_increment,
                            ndvi_contour_max,
                            ndvi_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-ndvi_contour_increment,
                            ndvi_contour_min,
                            -ndvi_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("NDVI value", breaks = seq(-1, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Normalised Difference Vegetation Index (NDVI)") +
  theme_classic() +
  theme(legend.position = "none")

ndvi_quad_3p
```

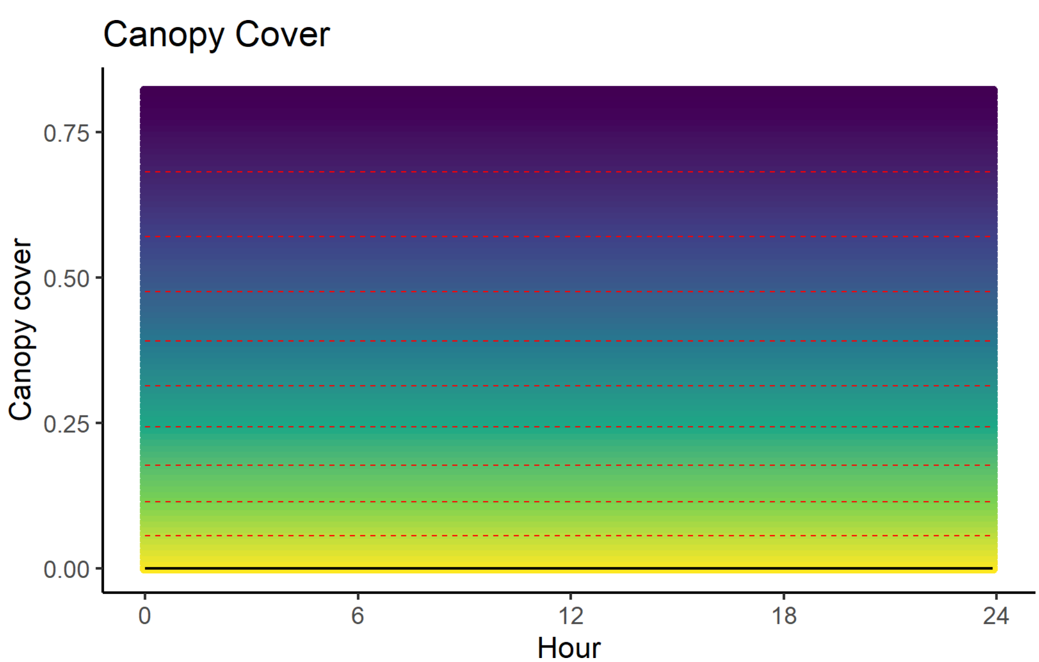


Canopy cover selection surface

0p

```
canopy_min <- min(buffalo_data$canopy_01, na.rm = TRUE)
canopy_max <- max(buffalo_data$canopy_01, na.rm = TRUE)
canopy_seq <- seq(canopy_min, canopy_max, by = 0.01)

# Create empty data frame
canopy_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_0p),
                                         nrow = length(canopy_seq)))

for(i in 1:nrow(hour_coefs_nat_df_0p)) {
  # Assign the vector as a column to the dataframe
  canopy_fresponse_df[,i] <- (hour_coefs_nat_df_0p$canopy[i] * canopy_seq) +
    (hour_coefs_nat_df_0p$canopy_2[i] * (canopy_seq ^ 2))
}

canopy_fresponse_df <- data.frame(canopy_seq, canopy_fresponse_df)
colnames(canopy_fresponse_df) <- c("canopy", hour)
canopy_fresponse_long <- pivot_longer(canopy_fresponse_df,
                                      cols = !1,
                                      names_to = "hour")

canopy_contour_min <- min(canopy_fresponse_long$value) # 0
```

```r
canopy_contour_max <- max(canopy_fresponse_long$value) # 2.181749
canopy_contour_increment <- (canopy_contour_max-canopy_contour_min)/10

canopy_quad_0p <- ggplot(data = canopy_fresponse_long, aes(x = as.numeric(hour),
                                                           y = canopy)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(canopy_contour_increment, canopy_contour_max,
                            -canopy_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
  breaks = seq(-canopy_contour_increment, canopy_contour_min,
               -canopy_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("Canopy cover", breaks = seq(0, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  ggtitle("Canopy Cover") +
  theme_classic() +
  theme(legend.position = "none")

canopy_quad_0p
```

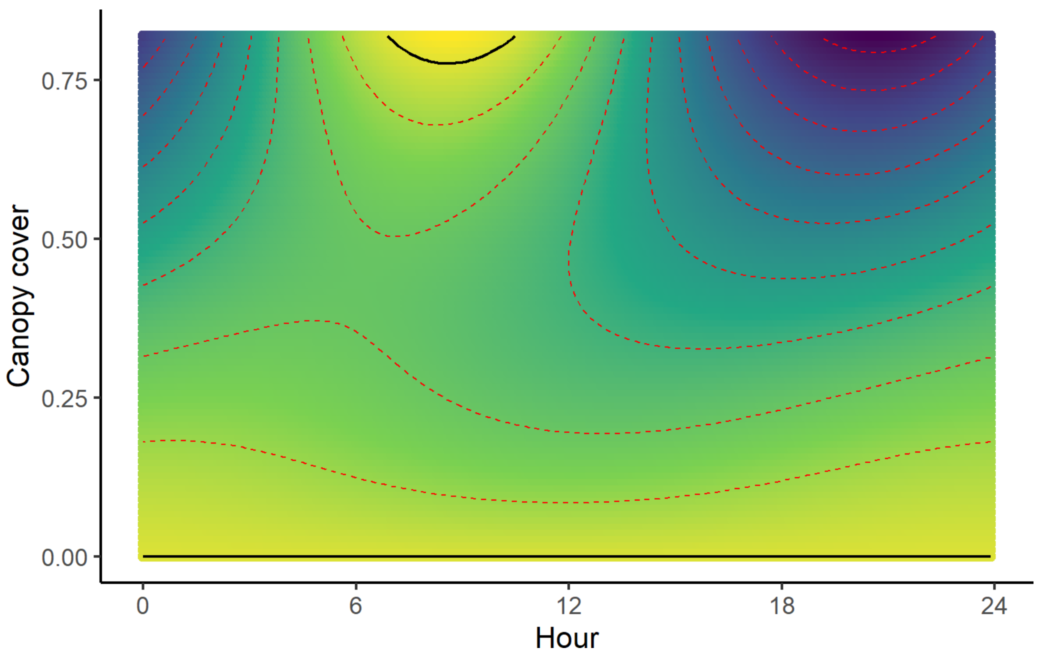


1p

```
canopy_min <- min(buffalo_data$canopy_01, na.rm = TRUE)
canopy_max <- max(buffalo_data$canopy_01, na.rm = TRUE)
canopy_seq <- seq(canopy_min, canopy_max, by = 0.01)

# Create empty data frame
canopy_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_1p),
                                         nrow = length(canopy_seq)))

for(i in 1:nrow(hour_coefs_nat_df_1p)) {
  # Assign the vector as a column to the dataframe
  canopy_fresponse_df[,i] <- (hour_coefs_nat_df_1p$canopy[i] * canopy_seq) +
    (hour_coefs_nat_df_1p$canopy_2[i] * (canopy_seq ^ 2))
}

canopy_fresponse_df <- data.frame(canopy_seq, canopy_fresponse_df)
colnames(canopy_fresponse_df) <- c("canopy", hour)
canopy_fresponse_long <- pivot_longer(canopy_fresponse_df, cols = !1,
                                      names_to = "hour")

canopy_contour_min <- min(canopy_fresponse_long$value) # 0
canopy_contour_max <- max(canopy_fresponse_long$value) # 2.181749
canopy_contour_increment <- (canopy_contour_max-canopy_contour_min)/10

canopy_quad_1p <- ggplot(data = canopy_fresponse_long,
                         aes(x = as.numeric(hour), y = canopy)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(canopy_contour_increment,
                            canopy_contour_max,
                            -canopy_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-canopy_contour_increment,
                            canopy_contour_min,
                            -canopy_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("Canopy cover", breaks = seq(0, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Canopy Cover") +
  theme_classic() +
```

```
  theme(legend.position = "none")

canopy_quad_1p
```

```
Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf
```



2p

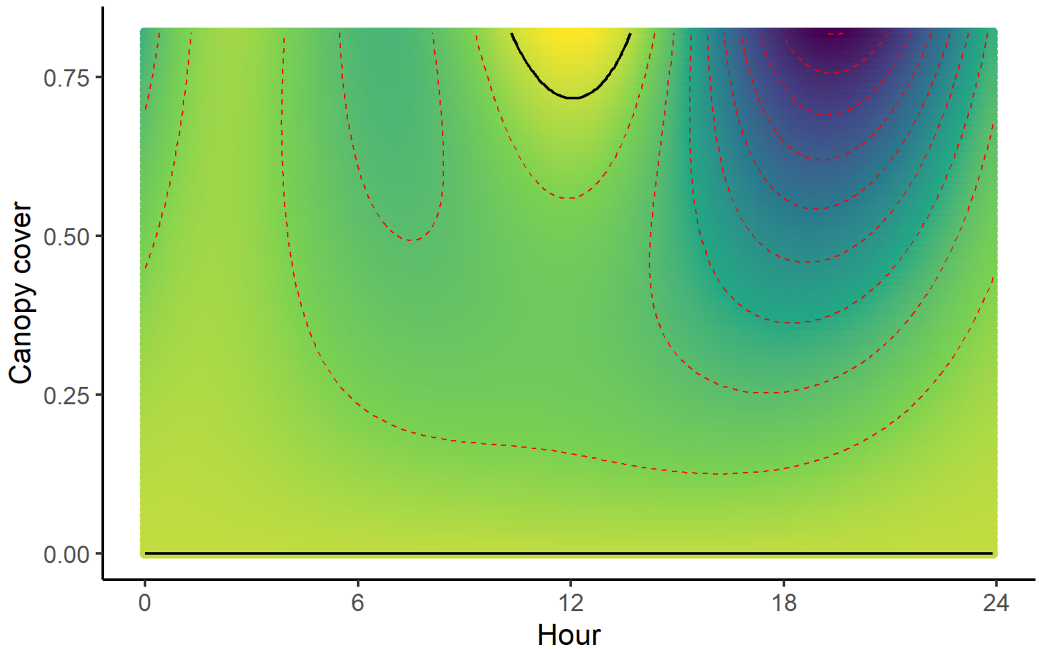
```
canopy_min <- min(buffalo_data$canopy_01, na.rm = TRUE)
canopy_max <- max(buffalo_data$canopy_01, na.rm = TRUE)
canopy_seq <- seq(canopy_min, canopy_max, by = 0.01)

# Create empty data frame
canopy_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_2p),
                                         nrow = length(canopy_seq)))

for(i in 1:nrow(hour_coefs_nat_df_2p)) {
  # Assign the vector as a column to the dataframe
  canopy_fresponse_df[,i] <- (hour_coefs_nat_df_2p$canopy[i] * canopy_seq) +
```

```r
    (hour_coefs_nat_df_2p$canopy_2[i] * (canopy_seq ^ 2))
}

canopy_fresponse_df <- data.frame(canopy_seq, canopy_fresponse_df)
colnames(canopy_fresponse_df) <- c("canopy", hour)
canopy_fresponse_long <- pivot_longer(canopy_fresponse_df, cols = !1,
                                      names_to = "hour")

canopy_contour_min <- min(canopy_fresponse_long$value) # 0
canopy_contour_max <- max(canopy_fresponse_long$value) # 2.181749
canopy_contour_increment <- (canopy_contour_max-canopy_contour_min)/10

canopy_quad_2p <- ggplot(data = canopy_fresponse_long,
                         aes(x = as.numeric(hour), y = canopy)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(canopy_contour_increment,
                            canopy_contour_max,
                            -canopy_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-canopy_contour_increment,
                            canopy_contour_min,
                            -canopy_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("Canopy cover", breaks = seq(0, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Canopy Cover") +
  theme_classic() +
  theme(legend.position = "none")

canopy_quad_2p
```

```
Warning: `stat_contour()`: Zero contours were generated
```

```
Warning in min(x): no non-missing arguments to min; returning Inf
```

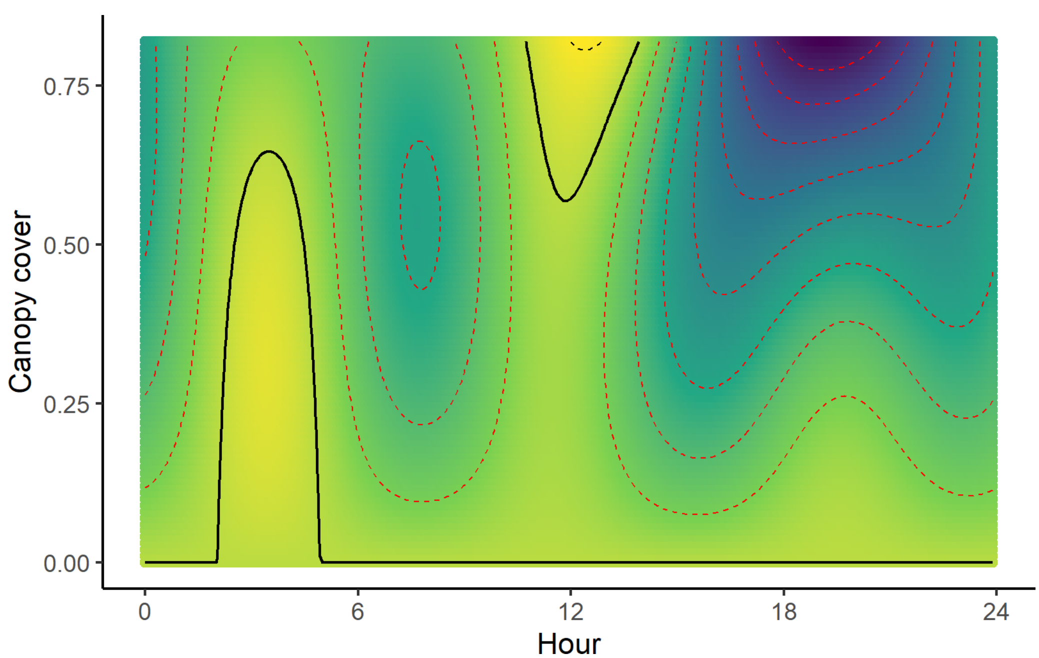
```
Warning in max(x): no non-missing arguments to max; returning -Inf
```



3p

```
canopy_min <- min(buffalo_data$canopy_01, na.rm = TRUE)
canopy_max <- max(buffalo_data$canopy_01, na.rm = TRUE)
canopy_seq <- seq(canopy_min, canopy_max, by = 0.01)

# Create empty data frame
canopy_fresponse_df <- data.frame(matrix(ncol = nrow(hour_coefs_nat_df_3p),
                                         nrow = length(canopy_seq)))

for(i in 1:nrow(hour_coefs_nat_df_3p)) {
  # Assign the vector as a column to the dataframe
  canopy_fresponse_df[,i] <- (hour_coefs_nat_df_3p$canopy[i] * canopy_seq) +
    (hour_coefs_nat_df_3p$canopy_2[i] * (canopy_seq ^ 2))
}

canopy_fresponse_df <- data.frame(canopy_seq, canopy_fresponse_df)
colnames(canopy_fresponse_df) <- c("canopy", hour)
canopy_fresponse_long <- pivot_longer(canopy_fresponse_df, cols = !1,
                                      names_to = "hour")

canopy_contour_min <- min(canopy_fresponse_long$value) # 0
canopy_contour_max <- max(canopy_fresponse_long$value) # 2.181749
canopy_contour_increment <- (canopy_contour_max-canopy_contour_min)/10
```

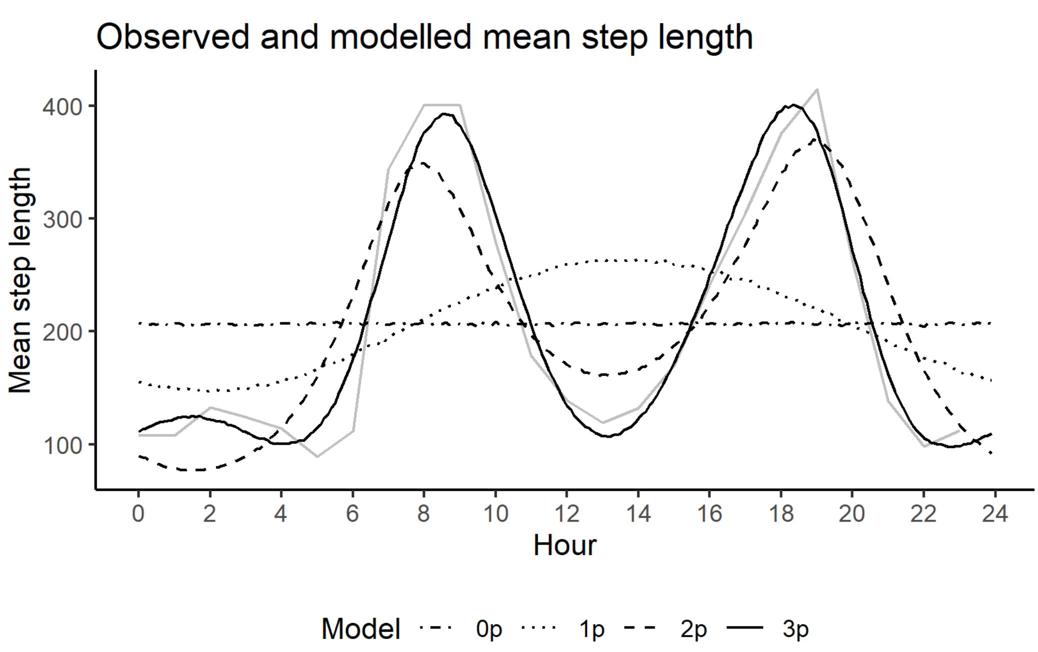
```r
canopy_quad_3p <- ggplot(data = canopy_fresponse_long,
                         aes(x = as.numeric(hour), y = canopy)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(canopy_contour_increment,
                            canopy_contour_max,
                            canopy_contour_increment),
               colour = "black", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value),
               breaks = seq(-canopy_contour_increment,
                            canopy_contour_min,
                            -canopy_contour_increment),
               colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("Canopy cover", breaks = seq(0, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  # ggtitle("Canopy Cover",
  #          subtitle = "Three pairs of harmonics") +
  theme_classic() +
  theme(legend.position = "none")

canopy_quad_3p
```



Combining the plots

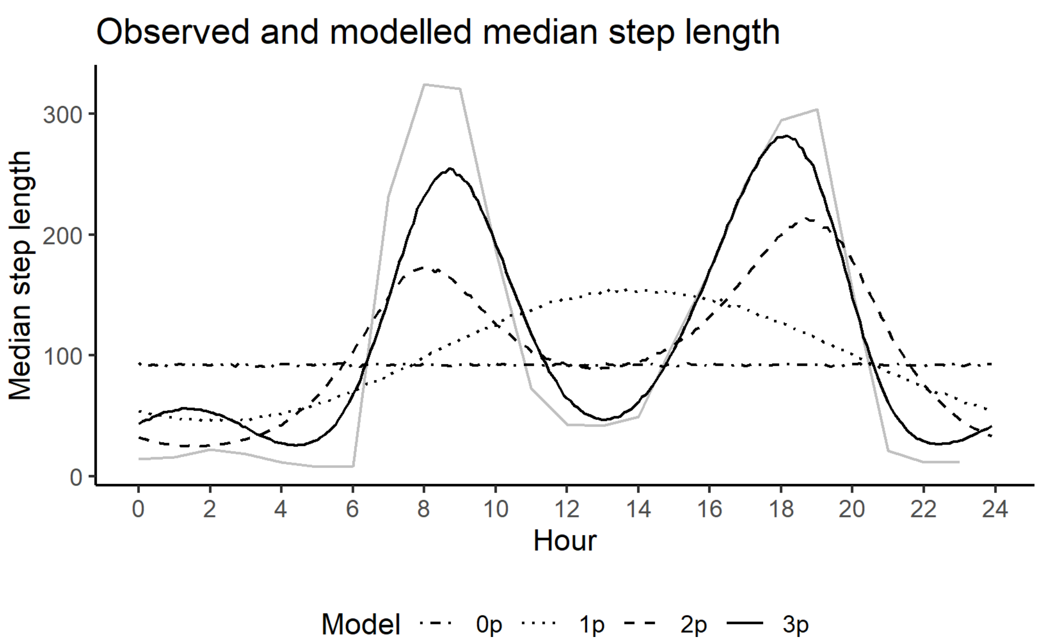
Movement parameters

```
gamma_df <- rbind(gamma_df_0p, gamma_df_1p, gamma_df_2p, gamma_df_3p)
gamma_df <- gamma_df %>% mutate(model_f = as.numeric(factor(model)))

mean_sl <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = mean_sl, group = factor(id)),
            alpha = 0.25) +
  geom_path(data = gamma_df, aes(x = hour, y = mean, linetype = model)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Mean step length") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
  ggtitle("Observed and modelled mean step length") +
  theme_classic() +
  theme(legend.position = "bottom")

mean_sl
```




```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/mean_sl_",
#           Sys.Date(), ".png"),
#   width=150, height=90, units="mm", dpi = 1000)
```

```
median_sl <- ggplot() +
  geom_path(data = movement_summary_buffalo,
            aes(x = hour, y = median_sl, group = factor(id)),
            alpha = 0.25) +
  geom_path(data = gamma_df, aes(x = hour, y = median, linetype = model)) +
  scale_x_continuous("Hour", breaks = seq(0,24,2)) +
  scale_y_continuous("Median step length") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
  ggtitle("Observed and modelled median step length") +
  theme_classic() +
  theme(legend.position = "bottom")

median_sl
```

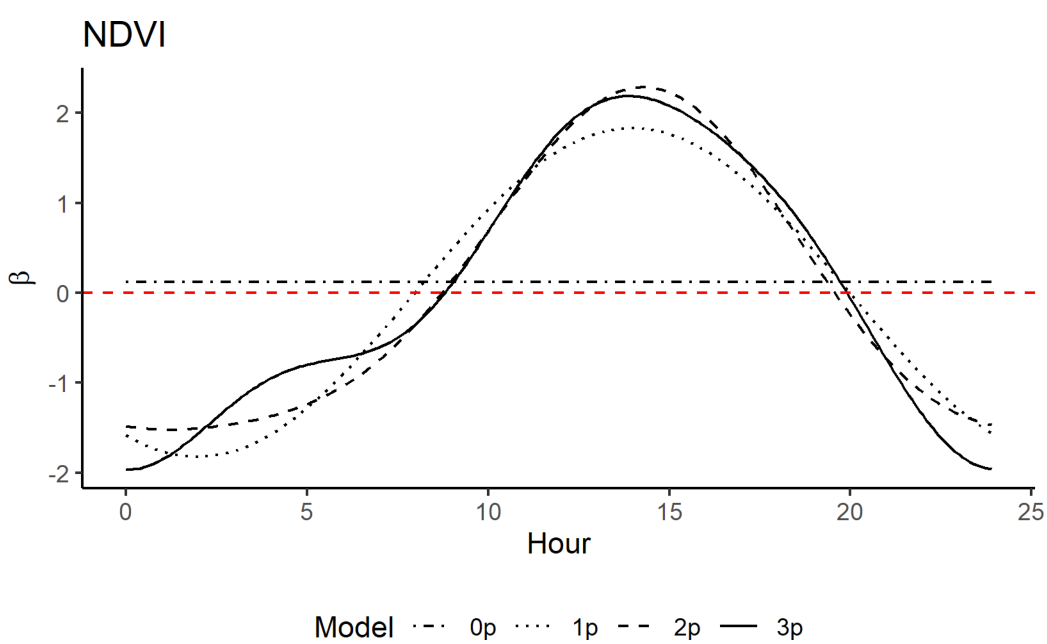



```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/median_sl_",
#          Sys.Date(), ".png"),
#   width=150, height=90, units="mm", dpi = 1000)
```

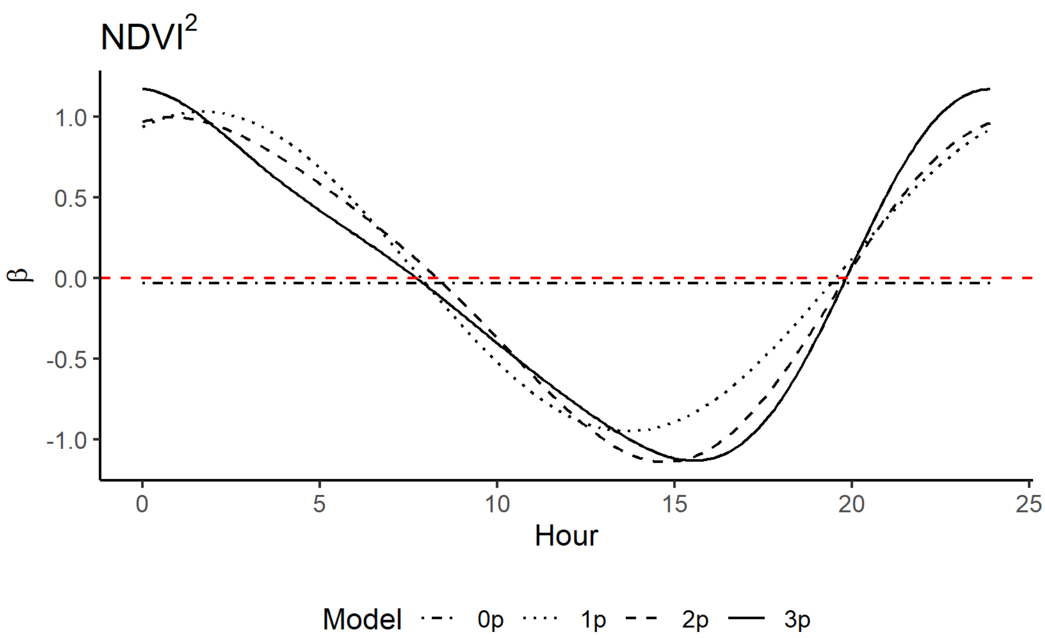
Habitat selection

```
harmonics_scaled_long_0p <- harmonics_scaled_long_0p %>% mutate(model = "0p")
harmonics_scaled_long_1p <- harmonics_scaled_long_1p %>% mutate(model = "1p")
```

```r
harmonics_scaled_long_2p <- harmonics_scaled_long_2p %>% mutate(model = "2p")
harmonics_scaled_long_3p <- harmonics_scaled_long_3p %>% mutate(model = "3p")

harmonics_scaled_long_Mp <- rbind(harmonics_scaled_long_0p,
                                  harmonics_scaled_long_1p,
                                  harmonics_scaled_long_2p,
                                  harmonics_scaled_long_3p)

coef_titles <- unique(harmonics_scaled_long_Mp$coef)


ndvi_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "ndvi"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle("NDVI") +
      theme_classic() +
      theme(legend.position = "bottom")

ndvi_harms
```

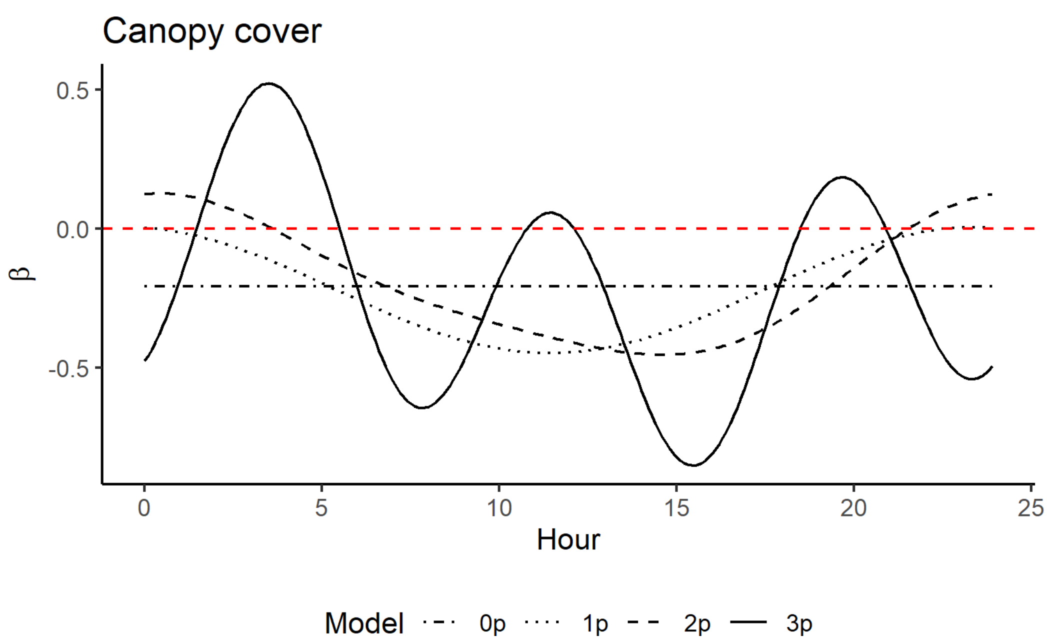


```r
ndvi_2_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "ndvi_2"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle(expression(NDVI^2)) +
      theme_classic() +
      theme(legend.position = "bottom")

ndvi_2_harms
```

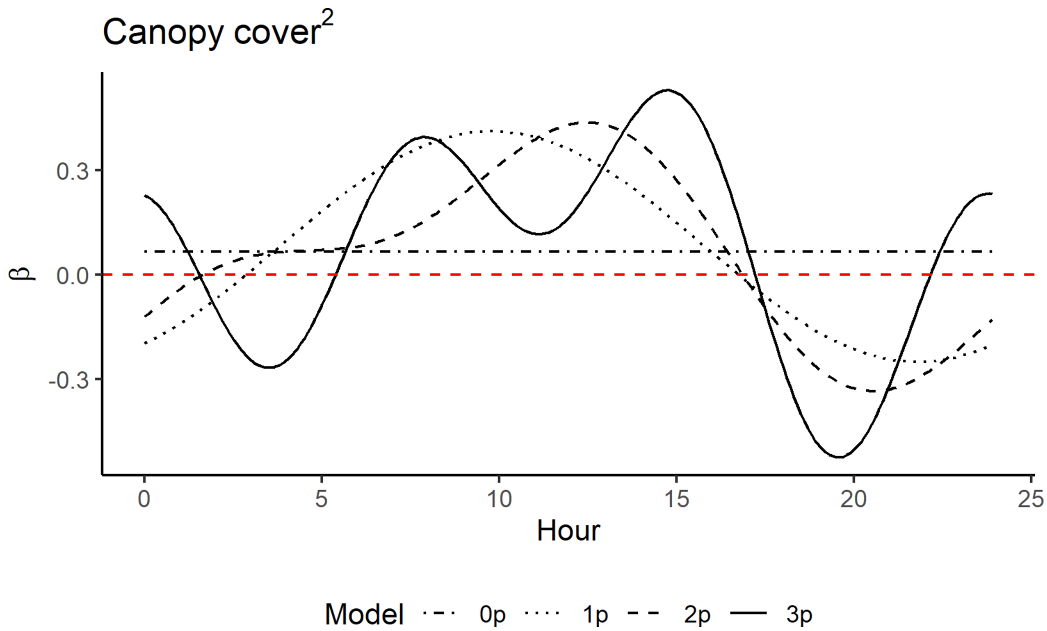



```r
canopy_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "canopy"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle("Canopy cover") +
      theme_classic() +
```

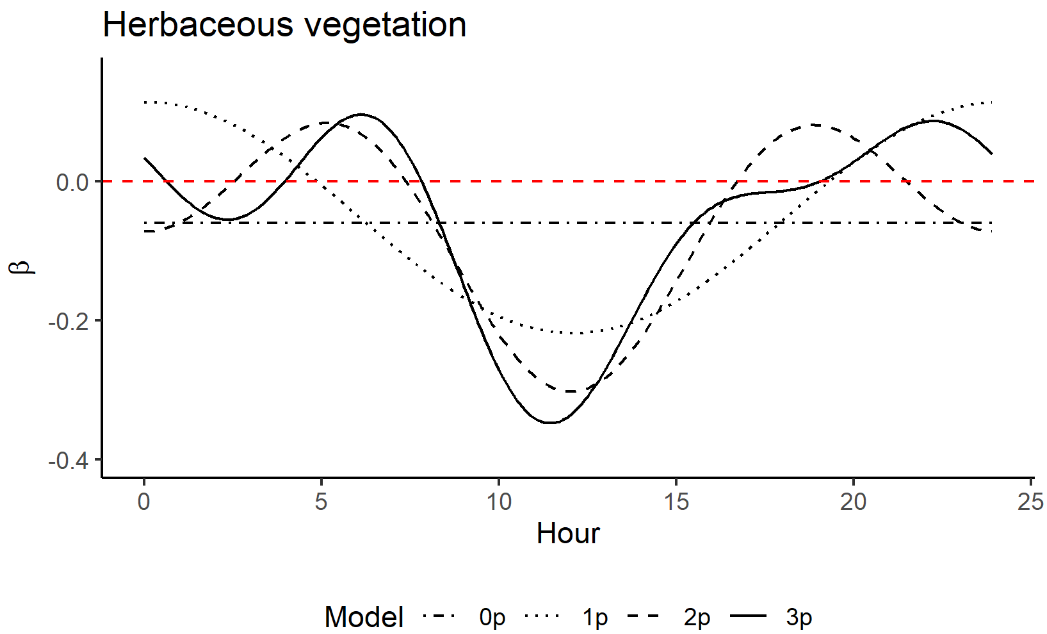
```
      theme(legend.position = "bottom")

canopy_harms
```



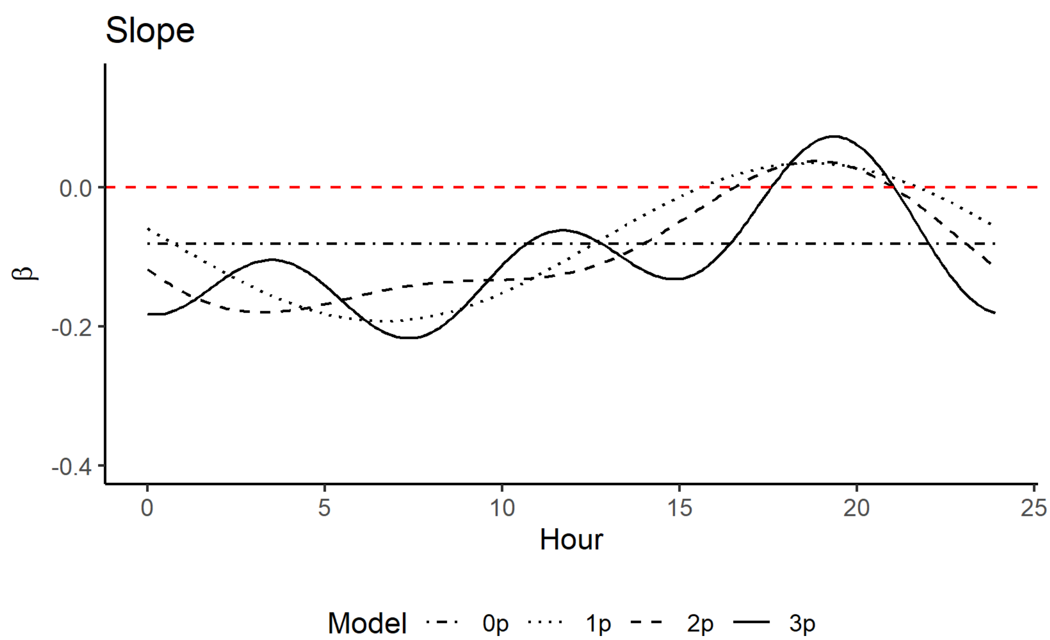

```
canopy_2_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "canopy_2"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle(expression(Canopy~cover^2)) +
      theme_classic() +
      theme(legend.position = "bottom")

canopy_2_harms
```



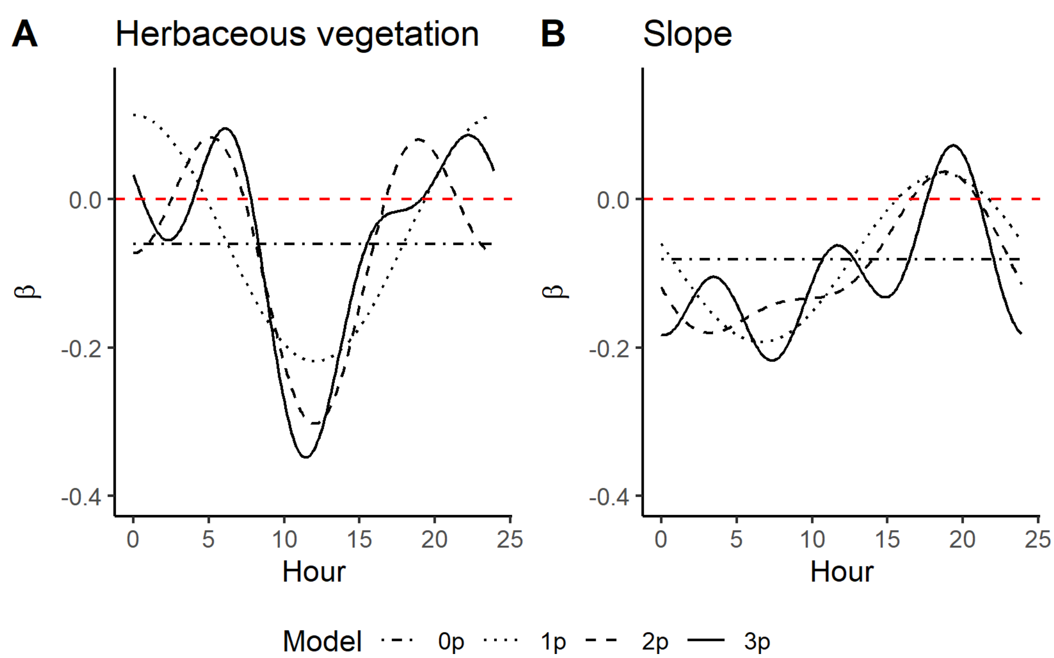

```
herby_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "herby"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta), limits = c(-0.4,0.15)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle("Herbaceous vegetation") +
      theme_classic() +
      theme(legend.position = "bottom")

herby_harms
```




```r
slope_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "slope"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta), limits = c(-0.4,0.15)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle("Slope") +
      theme_classic() +
      theme(legend.position = "bottom")

slope_harms
```




```r
ggarrange(herby_harms,
          slope_harms,
          labels = c("A", "B"),
          ncol = 2, nrow = 1,
          align = "hv",
          legend = "bottom",
          common.legend = TRUE)
```



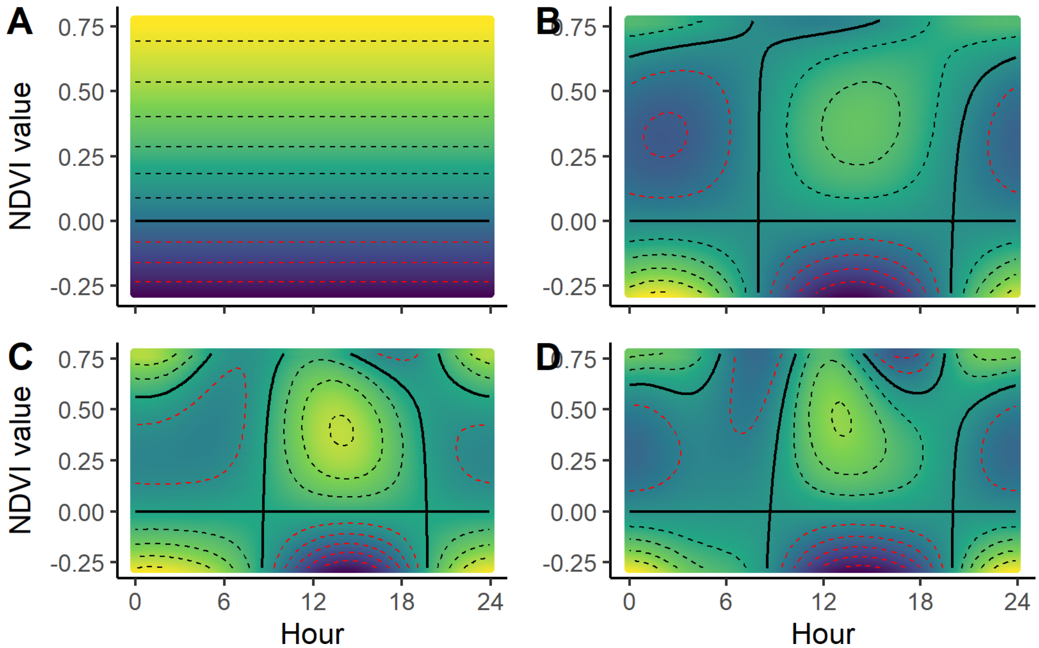
```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/herby_slope_harmonic_functions_",
#           Sys.Date(), ".png"),
#    width=150, height=90, units="mm", dpi = 1000)
```

Combining selection surfaces

NDVI

- A = 0p model
- B = 1p model
- C = 2p model
- D = 3p model

```
ggarrange(ndvi_quad_0p + theme(plot.title = element_blank(),
                               axis.title.x = element_blank(),
                               axis.text.x = element_blank()),

          ndvi_quad_1p + theme(plot.title = element_blank(),
                               axis.title.x = element_blank(),
                               axis.text.x = element_blank(),
                               axis.title.y = element_blank(),
                               ),

          ndvi_quad_2p,

          ndvi_quad_3p + theme(plot.title = element_blank(),
                               axis.title.y = element_blank(),
                               ),

          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2,
          legend = "none",
          common.legend = TRUE)
```




```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/",
#                "NDVI_2x2_CLR_TS_daily_GvM_10rs_",
#           Sys.Date(), ".png"),
#    width=150, height=120, units="mm", dpi = 1000)
```

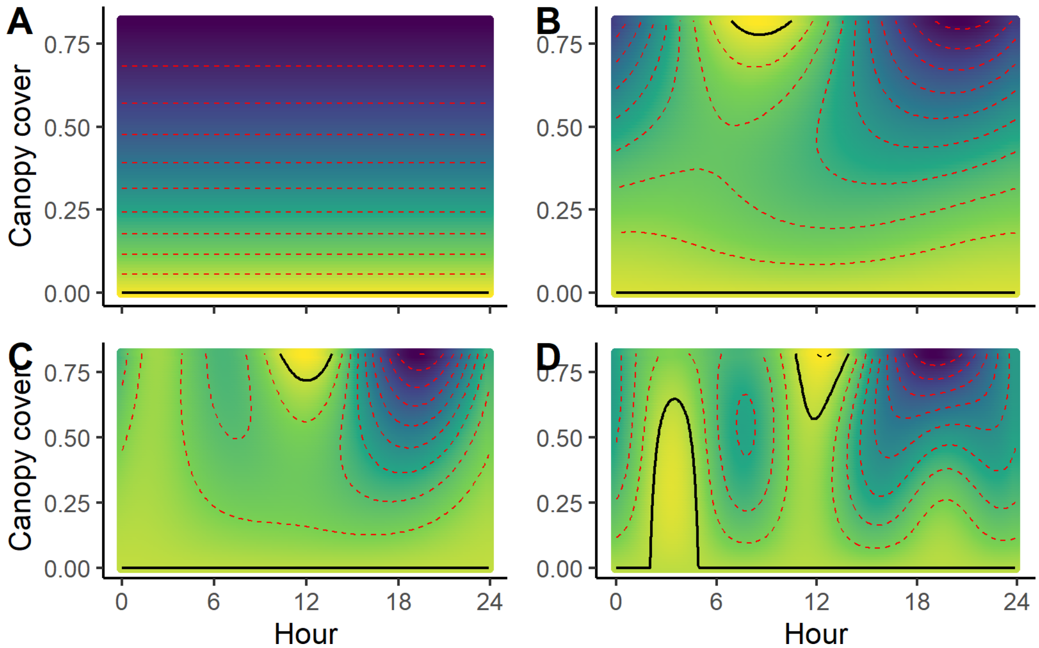
Canopy cover

- A = 0p model
- B = 1p model
- C = 2p model
- D = 3p model

```
ggarrange(canopy_quad_0p + theme(plot.title = element_blank(),
                                axis.title.x = element_blank(),
                                axis.text.x = element_blank()),

          canopy_quad_1p + theme(plot.title = element_blank(),
                                axis.title.x = element_blank(),
                                axis.text.x = element_blank(),
                                axis.title.y = element_blank(),
                                ),

          canopy_quad_2p,

          canopy_quad_3p + theme(plot.title = element_blank(),
```

```r
                              axis.title.y = element_blank(),
                              ),

          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2,
          legend = "none",
          common.legend = TRUE)
```

```
Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf

Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf

Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf

Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf
```




```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/",
#               "canopy_2x2_CLR_TS_daily_GvM_10rs_",
#          Sys.Date(), ".png"),
#   width=150, height=120, units="mm", dpi = 1000)
```

Adding all selection surfaces to the same plot

We combine these plots into the plot that is in the paper. On the top is the **NDVI** selection surface, and on the bottom is the **canopy cover** selection surface.

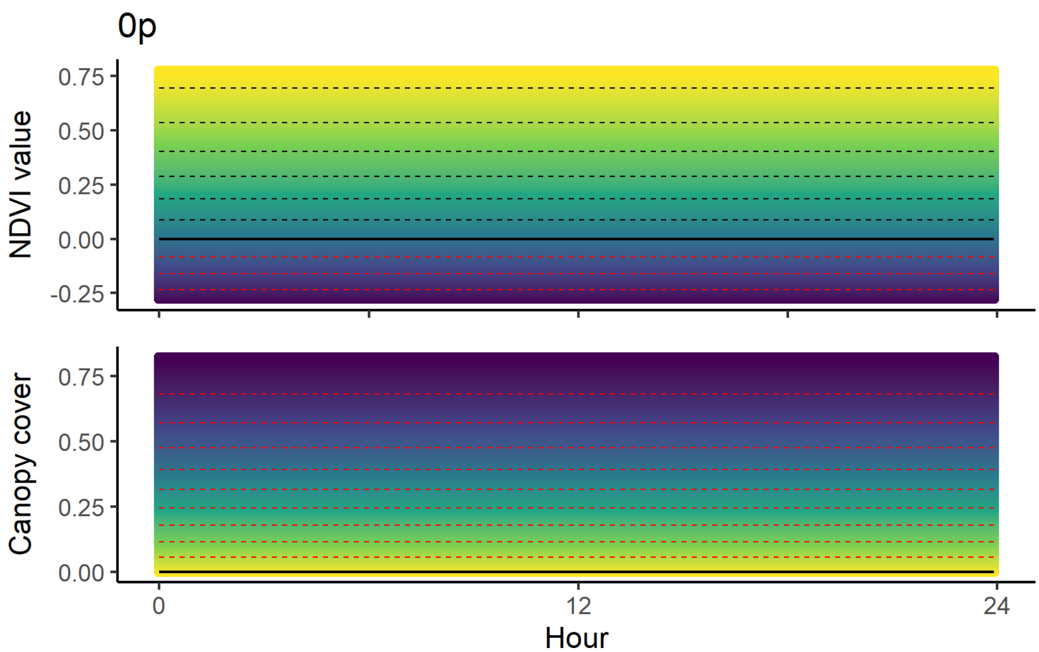
0p

```
surface_plots_0p <- ggarrange(ndvi_quad_0p +
            ggtitle("0p") +
            theme(axis.title.x = element_blank(),
                  axis.text.x = element_blank()),

          canopy_quad_0p +
            scale_x_continuous("Hour", breaks = c(0,12,24)) +
            theme(plot.title = element_blank()),

          ncol = 1, nrow = 2,
          align = "v",
```

```
          legend = "none",
          common.legend = TRUE)
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

```
surface_plots_0p
```



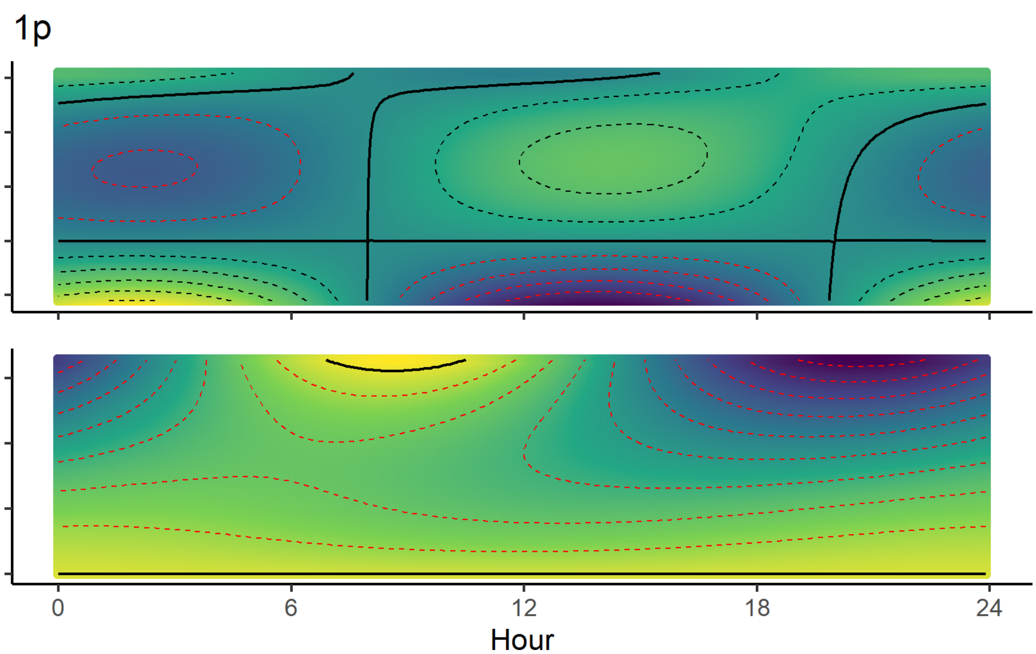
1p

```
surface_plots_1p <- ggarrange(ndvi_quad_1p +
            ggtitle("1p") +
            theme(axis.title.x = element_blank(),
                  axis.text.x = element_blank(),
                  axis.title.y = element_blank(),
                  axis.text.y = element_blank()),

          canopy_quad_1p +
            theme(plot.title = element_blank(),
                  axis.title.y = element_blank(),
                  axis.text.y = element_blank()),

          ncol = 1, nrow = 2,
```

```
         align = "v",
         legend = "none",
         common.legend = TRUE)
```

```
Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf

Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf
```

```
surface_plots_1p
```

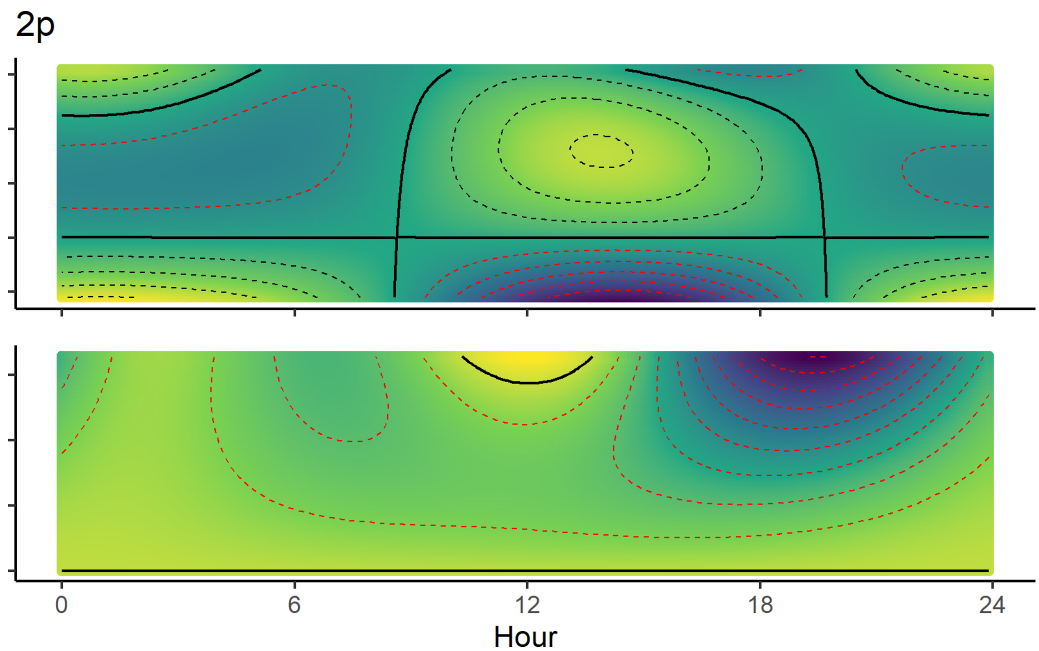


2p

```r
surface_plots_2p <- ggarrange(ndvi_quad_2p +
            ggtitle("2p") +
            theme(axis.title.x = element_blank(),
                  axis.text.x = element_blank(),
                  axis.title.y = element_blank(),
                  axis.text.y = element_blank()),

          canopy_quad_2p +
            theme(plot.title = element_blank(),
                  axis.title.y = element_blank(),
                  axis.text.y = element_blank()),

          ncol = 1, nrow = 2,
          align = "v",
          legend = "none",
          common.legend = TRUE)
```

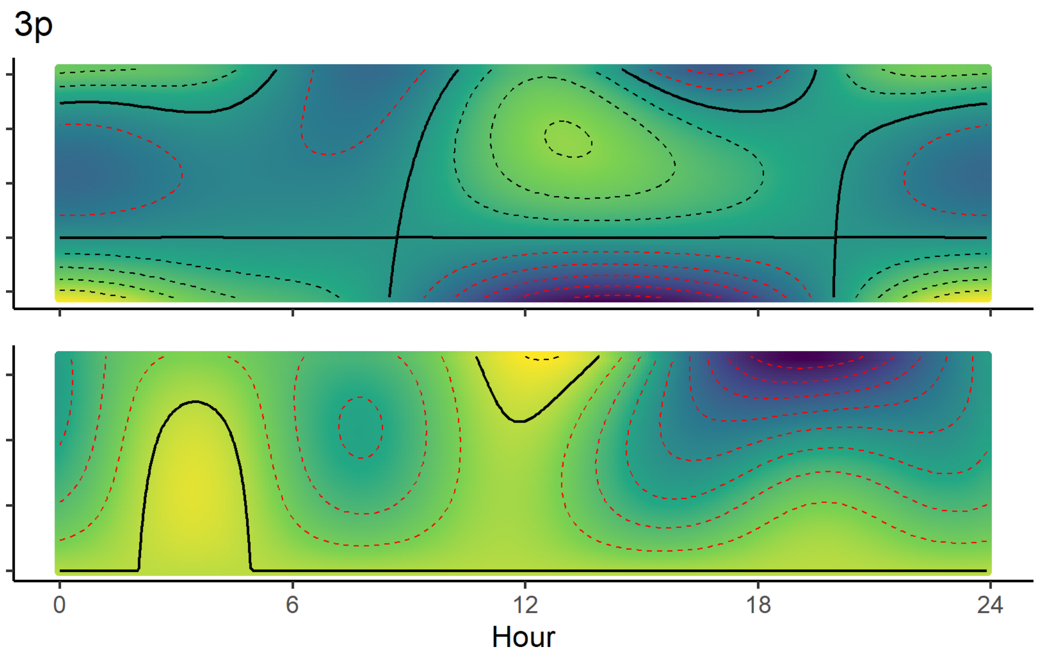
```
Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf

Warning: `stat_contour()`: Zero contours were generated

Warning in min(x): no non-missing arguments to min; returning Inf

Warning in max(x): no non-missing arguments to max; returning -Inf
```

```r
surface_plots_2p
```



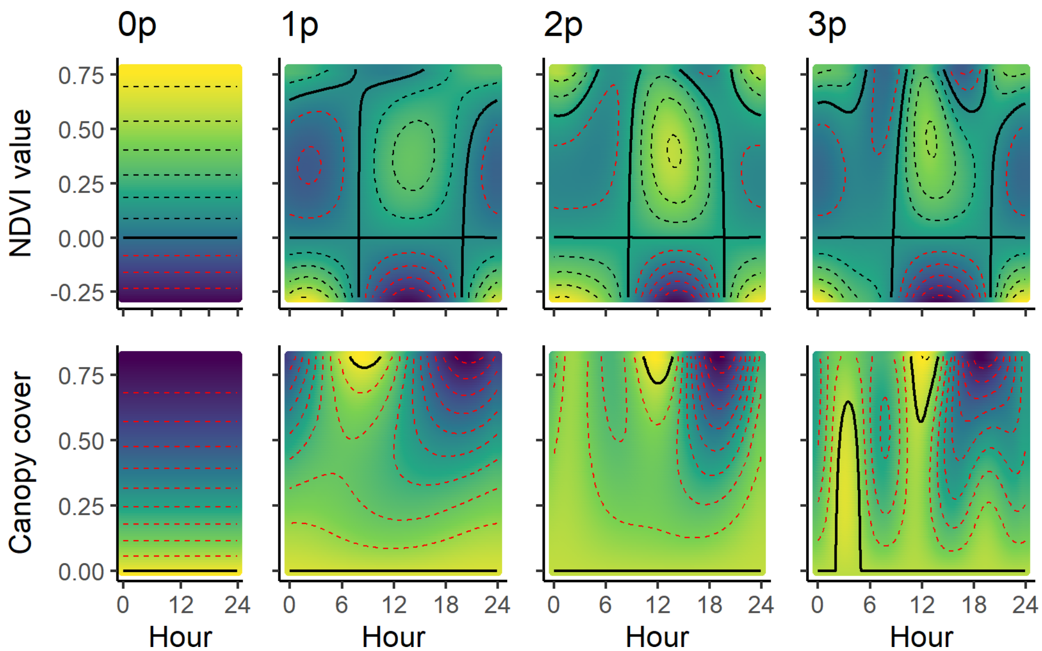
3p

```
surface_plots_3p <- ggarrange(ndvi_quad_3p +
            ggtitle("3p") +
              theme(axis.title.x = element_blank(),
                    axis.text.x = element_blank(),
                    axis.title.y = element_blank(),
                    axis.text.y = element_blank()),

           canopy_quad_3p +
            theme(plot.title = element_blank(),
                  axis.title.y = element_blank(),
                  axis.text.y = element_blank()),

          ncol = 1, nrow = 2,
          align = "v",
          legend = "none",
          common.legend = TRUE)

surface_plots_3p
```



All selection surfaces

```
all_selection_surfaces <- ggarrange(surface_plots_0p, surface_plots_1p, surface_plots_2p, su
          ncol = 4, nrow = 1
          # legend = "none",
          # legend.grob = get_legend(ndvi_quad_2p)
          )

all_selection_surfaces
```




```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/",
#               "all_quad_4x1_CLR_TS_daily_GvM_10rs_",
#          Sys.Date(), ".png"),
#   width=150, height=110, units="mm", dpi = 1000)
```

References

Fieberg, John, Johannes Signer, Brian Smith, and Tal Avgar. 2021. "A 'How to' Guide for Interpreting Parameters in Habitat-Selection Analyses." *The Journal of Animal Ecology* 90 (5): 1027–43. https://doi.org/10.1111/1365-2656.13441.

Forrest, Scott W, Dan Pagendam, Michael Bode, Christopher Drovandi, Jonathan R Potts, Justin Perry, Eric Vanderduys, and Andrew J Hoskins. 2024. "Predicting Fine-scale Distributions and Emergent Spatiotemporal Patterns from Temporally Dynamic Step Selection Simulations." *Ecography*, December. https://doi.org/10.1111/ecog.07421.

Session info

```
sessionInfo()
```

```
R version 4.4.1 (2024-06-14 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)
```

```
Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Brisbane
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] ggpubr_0.6.0    beepr_2.0       tictoc_1.2.1    terra_1.7-78
 [5] amt_0.2.2.0     lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
 [9] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
[13] tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.5       xfun_0.47          rstatix_0.7.2      lattice_0.22-6
 [5] tzdb_0.4.0         vctrs_0.6.5        tools_4.4.1        Rdpack_2.6.1
 [9] generics_0.1.3     parallel_4.4.1     proxy_0.4-27       fansi_1.0.6
[13] pkgconfig_2.0.3    Matrix_1.7-0       KernSmooth_2.23-24 lifecycle_1.0.4
[17] farver_2.1.2       compiler_4.4.1     munsell_0.5.1      tinytex_0.53
[21] codetools_0.2-20   carData_3.0-5      htmltools_0.5.8.1  class_7.3-22
[25] yaml_2.3.10        crayon_1.5.3       car_3.1-2          pillar_1.9.0
[29] classInt_0.4-10    magick_2.8.5       abind_1.4-8        tidyselect_1.2.1
[33] digest_0.6.37      stringi_1.8.4      sf_1.0-17          labeling_0.4.3
[37] splines_4.4.1      cowplot_1.1.3      fastmap_1.2.0      grid_4.4.1
[41] colorspace_2.1-1   cli_3.6.3          magrittr_2.0.3     survival_3.6-4
[45] utf8_1.2.4         broom_1.0.6        e1071_1.7-16       withr_3.0.1
[49] scales_1.3.0       backports_1.5.0    bit64_4.0.5        timechange_0.3.0
[53] rmarkdown_2.28     audio_0.1-11       bit_4.0.5          gridExtra_2.3
[57] ggsignif_0.6.4     hms_1.1.3          evaluate_1.0.0     knitr_1.48
[61] rbibutils_2.2.16   viridisLite_0.4.2  rlang_1.1.4        isoband_0.2.7
[65] Rcpp_1.0.13        glue_1.7.0         DBI_1.2.3          vroom_1.6.5
[69] jsonlite_1.8.8     R6_2.5.1           units_0.8-5
```