Fitting SSF models to the buffalo data

Dynamic step selection functions with temporal harmonics - wet season

Scott Forrest

2025-02-10

Table of contents

Load packages

```r
options(scipen=999)

library(tidyverse)
packages <- c("amt", "lubridate", "survival", "terra", "tictoc",
              "beepr", "ggpubr", "MASS", "patchwork", "scales")
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.

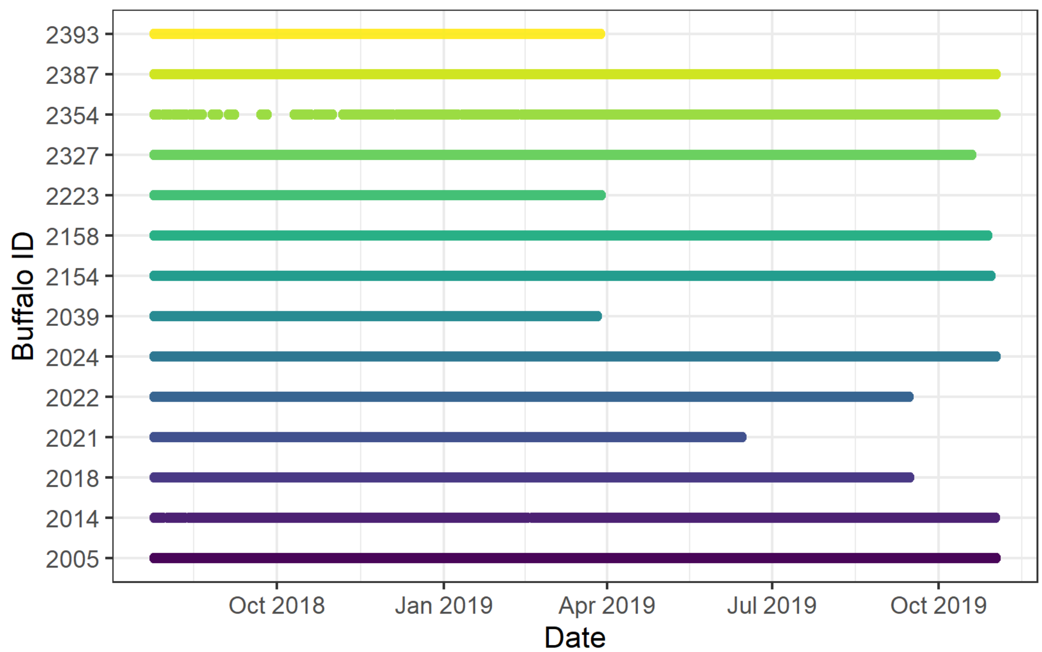
```r
buffalo_data_all <- read_csv("data/buffalo_parametric_popn_covs_GvM_10rs_2024-09-04.csv")
```

```
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.
```

```r
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))) +
```

```r
  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 se
# 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)
```

```r
# 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

```r
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,
  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 +
  step_l +
  log_step_l +
  cos_turn_a +

  strata(step_id)
```

1p

```
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_wet.rds"))) {

  model_0p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_0p_harms_wet.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_we

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

}
```

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

```
model_0p_harms
```

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

               coef exp(coef) se(coef)      z                  p
ndvi        0.31664   1.37250  0.07060  4.485  0.00000729038630065
ndvi_sq    -0.26853   0.76450  0.07620 -3.524             0.000425
canopy     -0.65865   0.51755  0.08418 -7.824  0.0000000000000512
canopy_sq   0.54151   1.71859  0.08199  6.604  0.00000000003989057
slope       0.01172   1.01179  0.02832  0.414             0.678946
herby       0.04652   1.04762  0.02628  1.771             0.076636
step_l     -0.28465   0.75227  0.03014 -9.444 < 0.0000000000000002
```

```
log_step_l  0.29081   1.33752  0.02866 10.147 < 0.0000000000000002
cos_turn_a  0.02667   1.02703  0.01814  1.470             0.141504

Likelihood ratio test=229.7  on 9 df, p=< 0.00000000000000022
n= 39660, number of events= 3390
   (864 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL

$more
NULL

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

1p

```
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_1p_harms_wet.rds"))) {

  model_1p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_1p_harms_wet.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_we

  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           0.27221   1.31287  0.09310   2.924             0.003456
ndvi_s1       -1.47785   0.22813  0.29048  -5.088 0.000000362757111271
ndvi_c1       -0.69794   0.49761  0.24641  -2.832             0.004620
ndvi_sq       -0.27254   0.76144  0.09374  -2.907             0.003646
ndvi_sq_s1     0.70234   2.01847  0.18185   3.862             0.000112
ndvi_sq_c1     0.35512   1.42635  0.15957   2.226             0.026046
canopy        -0.62154   0.53712  0.08764  -7.092 0.000000000001325872
canopy_s1      0.24421   1.27661  0.22771   1.072             0.283523
canopy_c1     -0.10677   0.89873  0.23101  -0.462             0.643951
canopy_sq      0.51114   1.66720  0.08511   6.006 0.000000001906791068
canopy_sq_s1   0.11711   1.12425  0.15777   0.742             0.457899
canopy_sq_c1   0.07700   1.08004  0.15768   0.488             0.625321
slope          0.01484   1.01495  0.02905   0.511             0.609436
slope_s1      -0.04321   0.95771  0.03778  -1.144             0.252717
slope_c1      -0.03944   0.96133  0.03931  -1.003             0.315737
herby          0.04505   1.04608  0.02745   1.641             0.100809
herby_s1      -0.01240   0.98768  0.05372  -0.231             0.817532
herby_c1       0.06057   1.06244  0.05728   1.057             0.290318
step_l        -0.31739   0.72805  0.03131 -10.138 < 0.0000000000000002
step_l_s1      0.03119   1.03168  0.03702   0.842             0.399536
step_l_c1      0.04169   1.04257  0.03735   1.116             0.264398
log_step_l     0.35845   1.43111  0.03111  11.520 < 0.0000000000000002
log_step_l_s1 -0.47356   0.62278  0.05880  -8.054 0.000000000000000804
log_step_l_c1 -0.18578   0.83045  0.05681  -3.270             0.001075
cos_turn_a     0.03057   1.03104  0.01844   1.657             0.097449
cos_turn_a_s1 -0.05759   0.94404  0.01843  -3.125             0.001779
cos_turn_a_c1  0.02252   1.02278  0.01867   1.206             0.227667

Likelihood ratio test=434  on 27 df, p=< 0.00000000000000022
n= 39660, number of events= 3390
   (864 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_wet.rds"))) {

  model_2p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_2p_harms_wet.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_we

  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        0.316097   1.371763  0.097318  3.248          0.001162
ndvi_s1    -1.420877   0.241502  0.301480 -4.713    0.0000024408584
ndvi_s2     0.205236   1.227815  0.273278  0.751          0.452643
ndvi_c1    -0.632995   0.530999  0.306605 -2.065          0.038967
ndvi_c2     0.170273   1.185629  0.285123  0.597          0.550379
```

```
ndvi_sq       -0.330068  0.718875  0.097580  -3.383             0.000718
ndvi_sq_s1     0.755142  2.127914  0.187450   4.029       0.0000561322922
ndvi_sq_s2     0.033228  1.033786  0.174182   0.191             0.848711
ndvi_sq_c1     0.292456  1.339713  0.190317   1.537             0.124372
ndvi_sq_c2     0.021747  1.021986  0.180771   0.120             0.904243
canopy        -0.579578  0.560135  0.089387  -6.484       0.0000000000894
canopy_s1      0.382994  1.466669  0.235389   1.627             0.103722
canopy_s2     -0.149778  0.860899  0.231294  -0.648             0.517266
canopy_c1      0.008450  1.008486  0.237985   0.036             0.971677
canopy_c2      0.158279  1.171493  0.235046   0.673             0.500696
canopy_sq      0.471813  1.602897  0.086603   5.448       0.0000000509435
canopy_sq_s1   0.046310  1.047399  0.162272   0.285             0.775348
canopy_sq_s2   0.113950  1.120697  0.159173   0.716             0.474058
canopy_sq_c1   0.011185  1.011248  0.161156   0.069             0.944666
canopy_sq_c2   0.005665  1.005681  0.160571   0.035             0.971858
slope          0.012820  1.012902  0.031345   0.409             0.682554
slope_s1      -0.047972  0.953160  0.040111  -1.196             0.231697
slope_s2      -0.122505  0.884702  0.041155  -2.977             0.002914
slope_c1      -0.038410  0.962318  0.042722  -0.899             0.368612
slope_c2       0.029798  1.030246  0.040788   0.731             0.465050
herby          0.031493  1.031994  0.028253   1.115             0.264983
herby_s1      -0.026990  0.973371  0.055660  -0.485             0.627739
herby_s2      -0.029304  0.971121  0.055920  -0.524             0.600253
herby_c1       0.031413  1.031912  0.059584   0.527             0.598045
herby_c2      -0.156702  0.854959  0.056753  -2.761             0.005761
step_l        -0.440720  0.643573  0.036136 -12.196 < 0.0000000000000002
step_l_s1      0.061805  1.063755  0.034710   1.781             0.074975
step_l_s2     -0.191891  0.825397  0.038767  -4.950       0.0000007428958
step_l_c1     -0.013252  0.986835  0.045126  -0.294             0.769004
step_l_c2     -0.212752  0.808357  0.040818  -5.212       0.0000001865672
log_step_l     0.443509  1.558165  0.033499  13.239 < 0.0000000000000002
log_step_l_s1 -0.636164  0.529319  0.067027  -9.491 < 0.0000000000000002
log_step_l_s2 -0.067998  0.934262  0.060114  -1.131             0.257991
log_step_l_c1 -0.107935  0.897686  0.059017  -1.829             0.067419
log_step_l_c2 -0.257057  0.773324  0.059509  -4.320       0.0000156299031
cos_turn_a     0.034114  1.034703  0.018741   1.820             0.068715
cos_turn_a_s1 -0.063351  0.938614  0.018750  -3.379             0.000728
cos_turn_a_s2 -0.054665  0.946802  0.018738  -2.917             0.003530
cos_turn_a_c1  0.027855  1.028247  0.018966   1.469             0.141903
cos_turn_a_c2 -0.057819  0.943821  0.018735  -3.086             0.002028

Likelihood ratio test=692.3  on 45 df, p=< 0.00000000000000022
n= 39660, number of events= 3390
   (864 observations deleted due to missingness)
```

```
$sl_
NULL

$ta_
NULL

$more
NULL

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

3p

```r
if(file.exists(paste0("ssf_coefficients/model_id", focal_id, "_3p_harms_wet.rds"))) {

  model_3p_harms <- readRDS(paste0("ssf_coefficients/model_id", focal_id, "_3p_harms_wet.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_we

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

}
```

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

```r
model_3p_harms
```

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

                   coef  exp(coef)  se(coef)       z                    p
```

ndvi	0.285568	1.330517	0.100048	2.854	0.00431
ndvi_s1	-1.390180	0.249031	0.331630	-4.192	0.00002765496513804
ndvi_s2	0.306752	1.359004	0.305604	1.004	0.31549
ndvi_s3	0.511608	1.667971	0.300996	1.700	0.08918
ndvi_c1	-0.764679	0.465483	0.293273	-2.607	0.00912
ndvi_c2	-0.278398	0.756995	0.320862	-0.868	0.38558
ndvi_c3	-0.432934	0.648603	0.269812	-1.605	0.10859
ndvi_sq	-0.293189	0.745881	0.100592	-2.915	0.00356
ndvi_sq_s1	0.741018	2.098071	0.203963	3.633	0.00028
ndvi_sq_s2	-0.055430	0.946078	0.189001	-0.293	0.76931
ndvi_sq_s3	-0.418150	0.658264	0.189239	-2.210	0.02713
ndvi_sq_c1	0.422817	1.526256	0.185085	2.284	0.02235
ndvi_sq_c2	0.342438	1.408377	0.200928	1.704	0.08833
ndvi_sq_c3	0.196873	1.217589	0.174117	1.131	0.25818
canopy	-0.583838	0.557754	0.091736	-6.364	0.00000000019610682
canopy_s1	0.380917	1.463626	0.240952	1.581	0.11390
canopy_s2	-0.197028	0.821167	0.238089	-0.828	0.40793
canopy_s3	-0.052420	0.948930	0.238284	-0.220	0.82588
canopy_c1	-0.073932	0.928734	0.242548	-0.305	0.76051
canopy_c2	0.061447	1.063374	0.245770	0.250	0.80257
canopy_c3	-0.365150	0.694092	0.237933	-1.535	0.12486
canopy_sq	0.478132	1.613059	0.088702	5.390	0.00000007033367032
canopy_sq_s1	0.043557	1.044519	0.165976	0.262	0.79299
canopy_sq_s2	0.124074	1.132100	0.163257	0.760	0.44726
canopy_sq_s3	-0.046423	0.954638	0.163841	-0.283	0.77692
canopy_sq_c1	0.079051	1.082259	0.164240	0.481	0.63030
canopy_sq_c2	0.069921	1.072423	0.167190	0.418	0.67579
canopy_sq_c3	0.153049	1.165382	0.162233	0.943	0.34548
slope	-0.006339	0.993681	0.032245	-0.197	0.84415
slope_s1	-0.039502	0.961268	0.041157	-0.960	0.33716
slope_s2	-0.113396	0.892797	0.041491	-2.733	0.00628
slope_s3	0.021763	1.022002	0.041989	0.518	0.60424
slope_c1	-0.024348	0.975946	0.043441	-0.560	0.57515
slope_c2	0.010183	1.010235	0.043000	0.237	0.81280
slope_c3	-0.075361	0.927409	0.041887	-1.799	0.07200
herby	0.024226	1.024522	0.028817	0.841	0.40053
herby_s1	-0.012368	0.987708	0.056910	-0.217	0.82796
herby_s2	-0.026012	0.974323	0.058148	-0.447	0.65463
herby_s3	-0.002061	0.997941	0.058508	-0.035	0.97189
herby_c1	0.055623	1.057199	0.061006	0.912	0.36189
herby_c2	-0.134127	0.874479	0.059428	-2.257	0.02401
herby_c3	0.111934	1.118439	0.056483	1.982	0.04751
step_l	-0.572028	0.564380	0.040169	-14.240	< 0.0000000000000002
step_l_s1	0.193827	1.213887	0.042095	4.605	0.00000413339915260
step_l_s2	-0.216874	0.805031	0.042036	-5.159	0.00000024791556794

```
step_l_s3      0.062735  1.064744  0.043207   1.452              0.14651
step_l_c1      0.132521  1.141703  0.049154   2.696              0.00702
step_l_c2     -0.144700  0.865282  0.044655  -3.240              0.00119
step_l_c3     -0.026849  0.973508  0.042470  -0.632              0.52726
log_step_l     0.630235  1.878051  0.039754  15.853 < 0.0000000000000002
log_step_l_s1 -0.922517  0.397517  0.082748 -11.149 < 0.0000000000000002
log_step_l_s2  0.062840  1.064856  0.065413   0.961              0.33672
log_step_l_s3  0.788434  2.199948  0.065333  12.068 < 0.0000000000000002
log_step_l_c1 -0.349929  0.704738  0.058664  -5.965  0.00000000244639337
log_step_l_c2 -0.653859  0.520035  0.071615  -9.130 < 0.0000000000000002
log_step_l_c3  0.395543  1.485191  0.063387   6.240  0.00000000043725265
cos_turn_a     0.023868  1.024155  0.018997   1.256              0.20898
cos_turn_a_s1 -0.057512  0.944111  0.019158  -3.002              0.00268
cos_turn_a_s2 -0.041156  0.959679  0.018970  -2.170              0.03004
cos_turn_a_s3  0.151176  1.163202  0.019089   7.920  0.00000000000000238
cos_turn_a_c1  0.010286  1.010339  0.019168   0.537              0.59153
cos_turn_a_c2 -0.077024  0.925868  0.019257  -4.000  0.00006339000271146
cos_turn_a_c3 -0.007429  0.992599  0.018872  -0.394              0.69386

Likelihood ratio test=1131  on 63 df, p=< 0.00000000000000022
n= 39660, number of events= 3390
   (864 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        0.31664   1.37250  0.07060  4.485  0.00000729038630065
ndvi_sq    -0.26853   0.76450  0.07620 -3.524             0.000425
canopy     -0.65865   0.51755  0.08418 -7.824  0.00000000000000512
canopy_sq   0.54151   1.71859  0.08199  6.604  0.00000000003989057
slope       0.01172   1.01179  0.02832  0.414             0.678946
herby       0.04652   1.04762  0.02628  1.771             0.076636
step_l     -0.28465   0.75227  0.03014 -9.444 < 0.0000000000000002
log_step_l  0.29081   1.33752  0.02866 10.147 < 0.0000000000000002
cos_turn_a  0.02667   1.02703  0.01814  1.470             0.141504

Likelihood ratio test=229.7  on 9 df, p=< 0.00000000000000022
n= 39660, number of events= 3390
   (864 observations deleted due to missingness)

$sl_
NULL

$ta_
NULL

$more
NULL

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

```
# 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
```

```r
# 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)
coefs_clr_0p$scale_sd <- scaling_attributes_0p$sd
coefs_clr_0p <- coefs_clr_0p %>% mutate(value_nat = value / scale_sd)
head(coefs_clr_0p)
```

```
              coefs       value   scale_sd   value_nat
ndvi           ndvi  0.31663561 0.10522124  3.00923665
ndvi_sq     ndvi_sq -0.26853394 0.06541644 -4.10499172
canopy       canopy -0.65864839 0.17178048 -3.83424474
canopy_sq canopy_sq  0.54150612 0.13152570  4.11711252
slope         slope  0.01171978 0.60654424  0.01932222
herby         herby  0.04652146 0.42516832  0.10941893
```

1p

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

# 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)
coefs_clr_1p$scale_sd <- scaling_attributes_1p$sd
coefs_clr_1p <- coefs_clr_1p %>% mutate(value_nat = value / scale_sd)
head(coefs_clr_1p)
```

```
                coefs      value   scale_sd value_nat
ndvi             ndvi  0.2722137 0.10522124  2.587061
ndvi_s1       ndvi_s1 -1.4778477 0.23444702 -6.303546
ndvi_c1       ndvi_c1 -0.6979364 0.23199203 -3.008450
ndvi_sq       ndvi_sq -0.2725404 0.06541644 -4.166238
ndvi_sq_s1 ndvi_sq_s1  0.7023402 0.08930182  7.864792
ndvi_sq_c1 ndvi_sq_c1  0.3551197 0.09024920  3.934879
```

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)
coefs_clr_2p$scale_sd <- scaling_attributes_2p$sd
coefs_clr_2p <- coefs_clr_2p %>% mutate(value_nat = value / scale_sd)
head(coefs_clr_2p)
```

```
          coefs      value   scale_sd  value_nat
ndvi       ndvi  0.3160969 0.10522124  3.0041168
ndvi_s1 ndvi_s1 -1.4208772 0.23444702 -6.0605471
ndvi_s2 ndvi_s2  0.2052364 0.22900591  0.8962059
ndvi_c1 ndvi_c1 -0.6329950 0.23199203 -2.7285206
ndvi_c2 ndvi_c2  0.1702731 0.23717765  0.7179140
ndvi_sq ndvi_sq -0.3300680 0.06541644 -5.0456435
```

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)
coefs_clr_3p$scale_sd <- scaling_attributes_3p$sd
```

```
coefs_clr_3p <- coefs_clr_3p %>% mutate(value_nat = value / scale_sd)
head(coefs_clr_3p)
```

```
          coefs      value  scale_sd value_nat
ndvi       ndvi  0.2855679 0.1052212  2.713975
ndvi_s1 ndvi_s1 -1.3901797 0.2344470 -5.929611
ndvi_s2 ndvi_s2  0.3067524 0.2290059  1.339495
ndvi_s3 ndvi_s3  0.5116079 0.2310828  2.213959
ndvi_c1 ndvi_c1 -0.7646792 0.2319920 -3.296144
ndvi_c2 ndvi_c2 -0.2783981 0.2371776 -1.173796
```

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))),
```

```r
  "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

```
# 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(
```

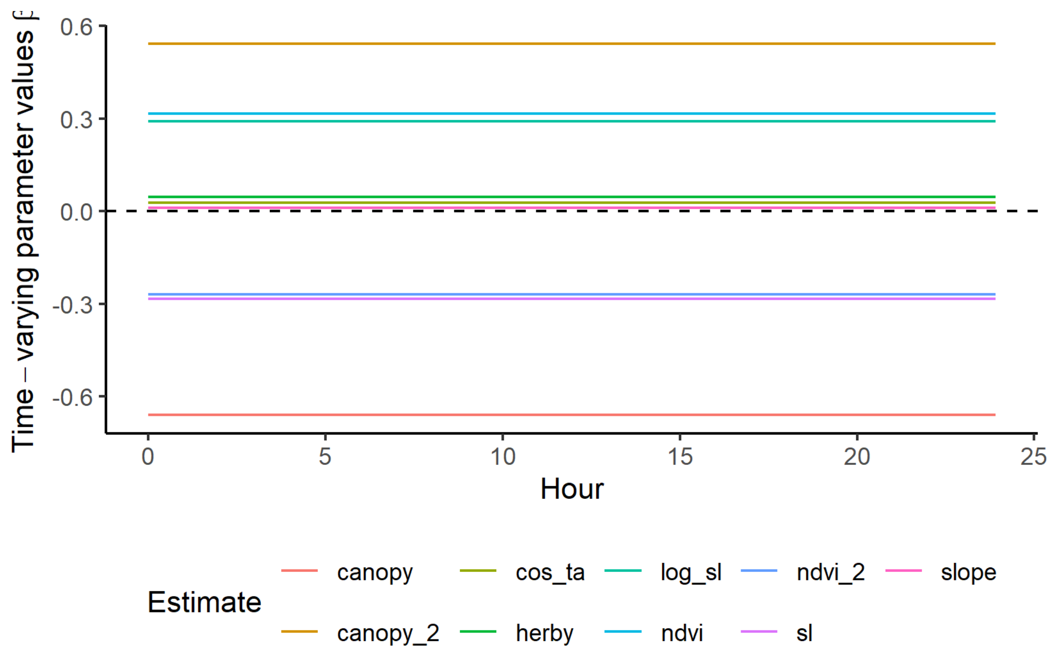
```
    coefs_clr_3p %>% dplyr::filter(grepl("step_l", coefs) & !grepl("log", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "log_sl" = as.numeric(
    coefs_clr_3p %>% dplyr::filter(grepl("log_step_l", coefs)) %>%
      pull(value) %>% t() %*% t(as.matrix(hour_harmonics_df_3p))),
  "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).

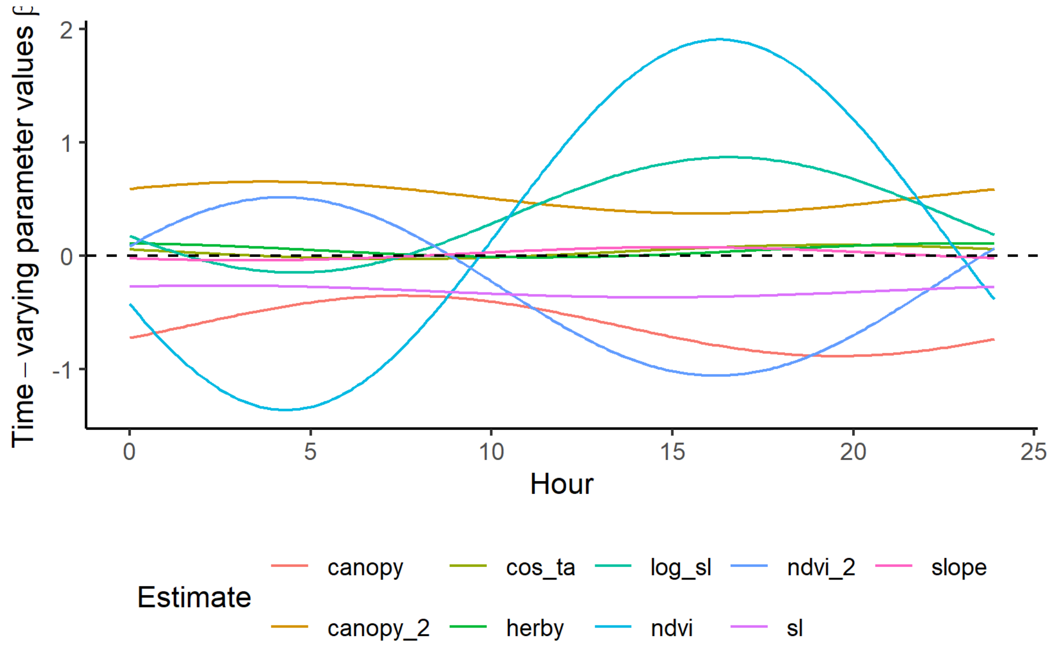
0p

```
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")
```



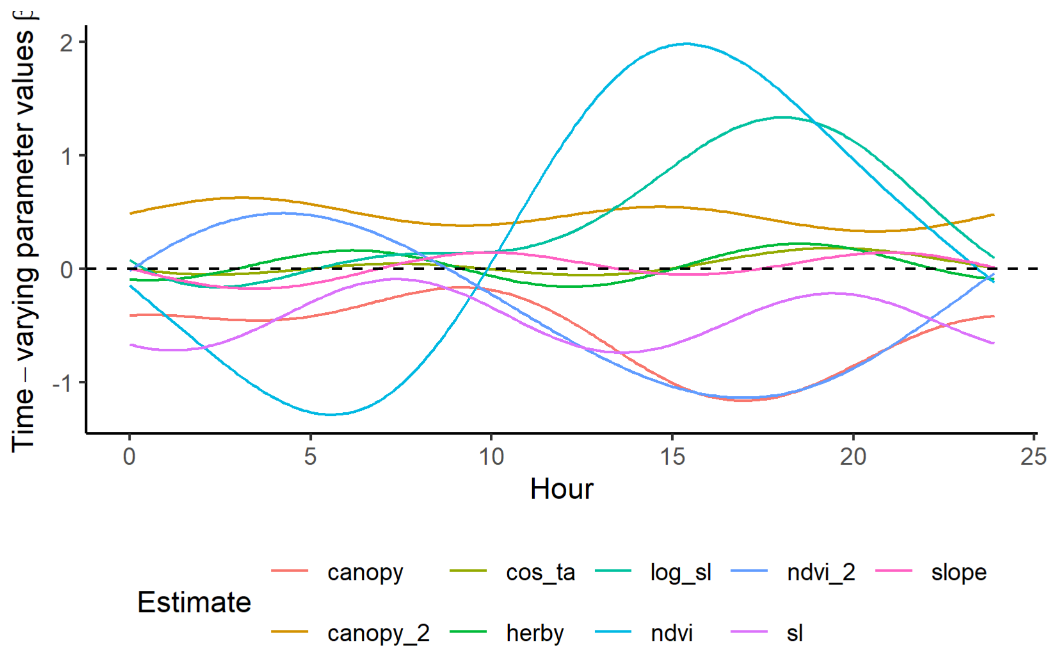
1p

```r
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")
```



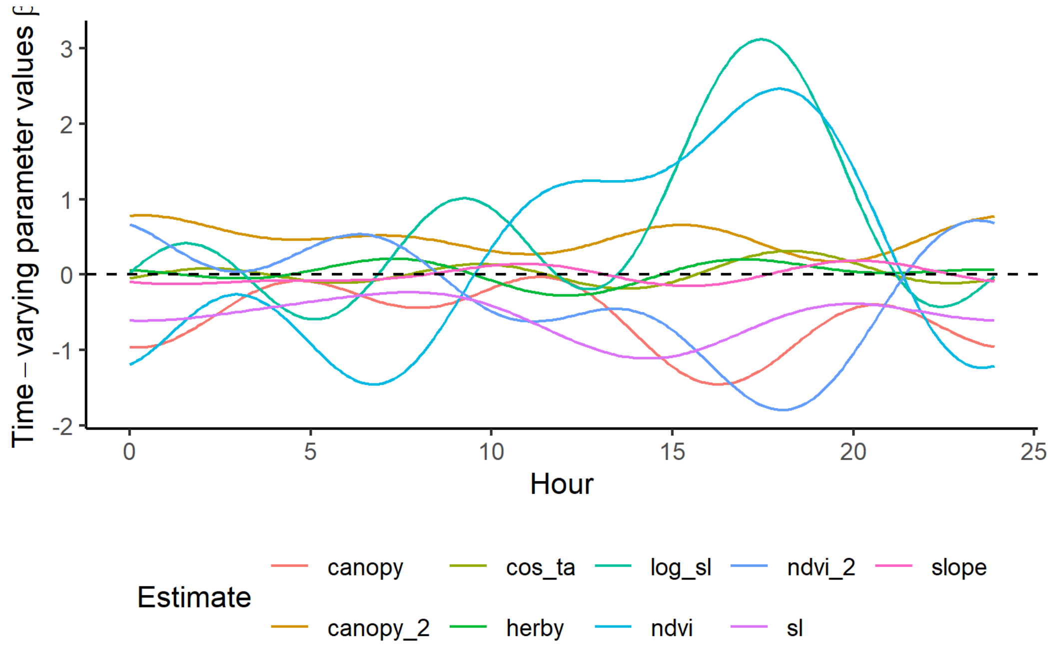
2p

```
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

```r
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,
```

```r
         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_wet_", 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

```r
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_wet_",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

```r
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_wet_",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_wet_", 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")
```

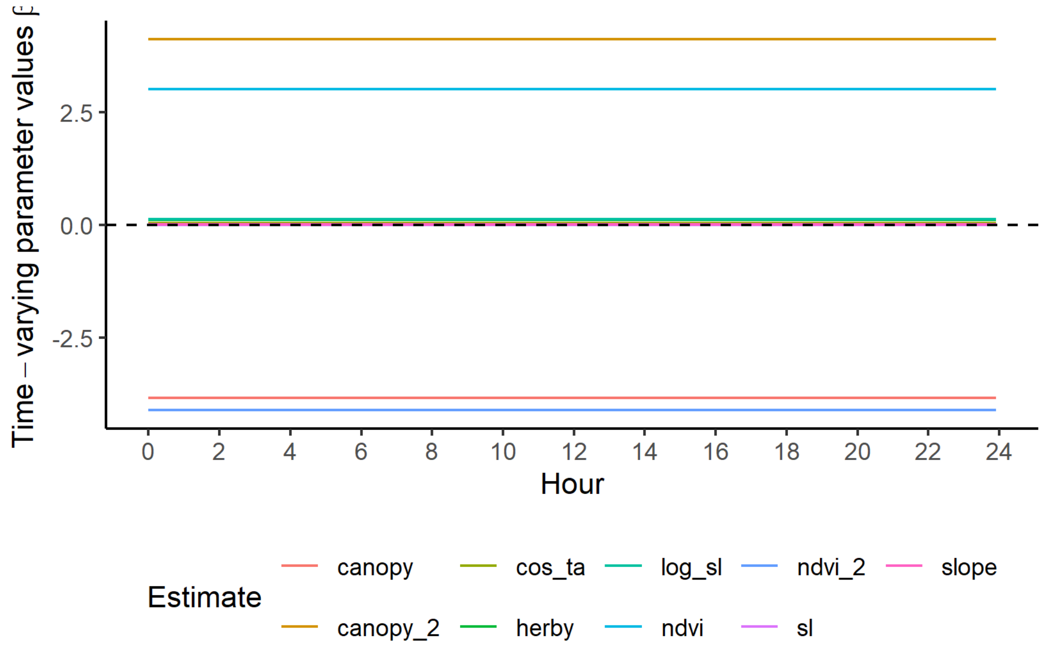
Plot the natural-scale temporally dynamic coefficients

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.

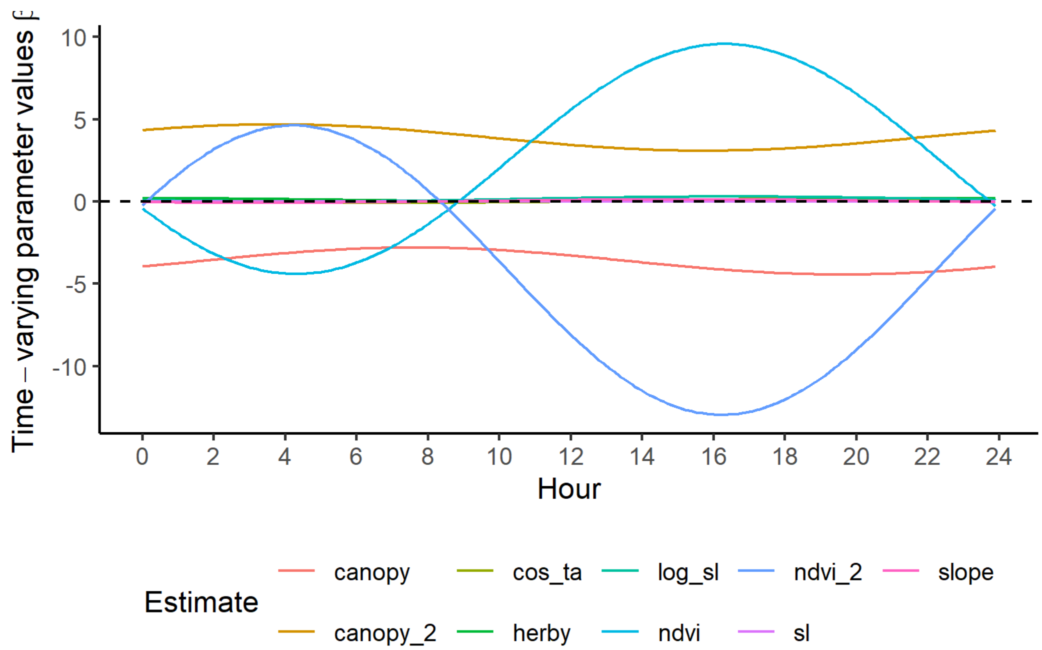
0p

```
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")
```



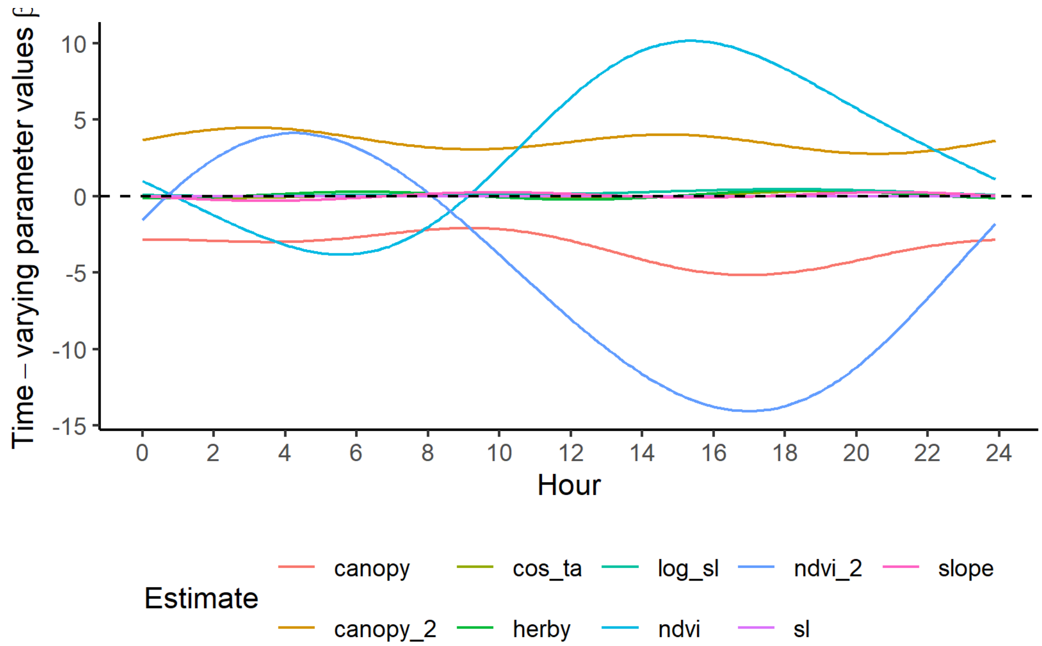
1p

```
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")
```



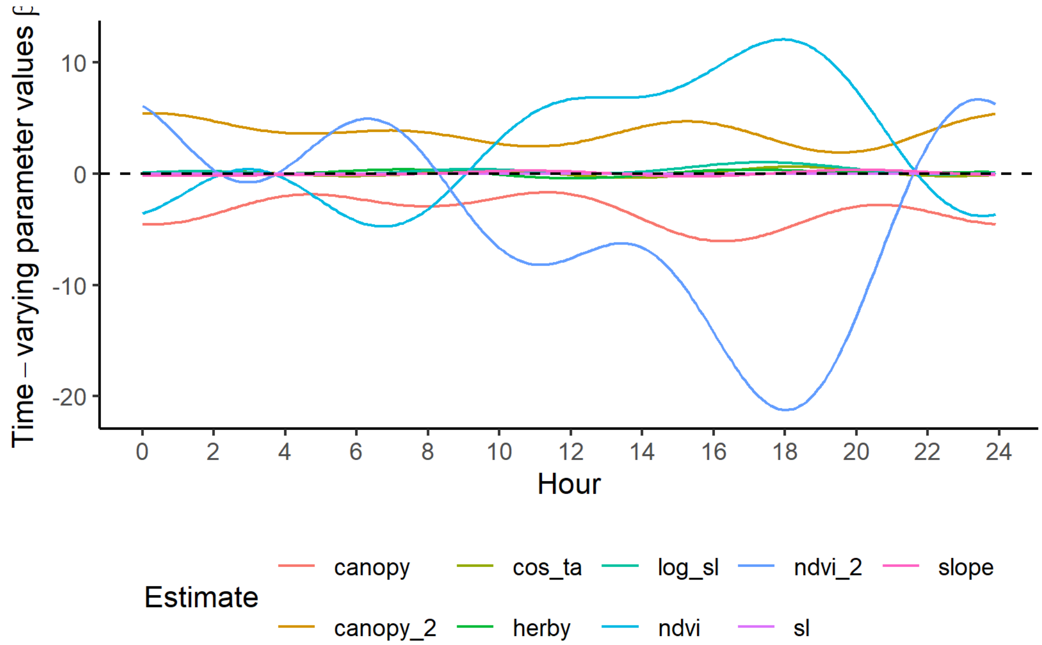
2p

```
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

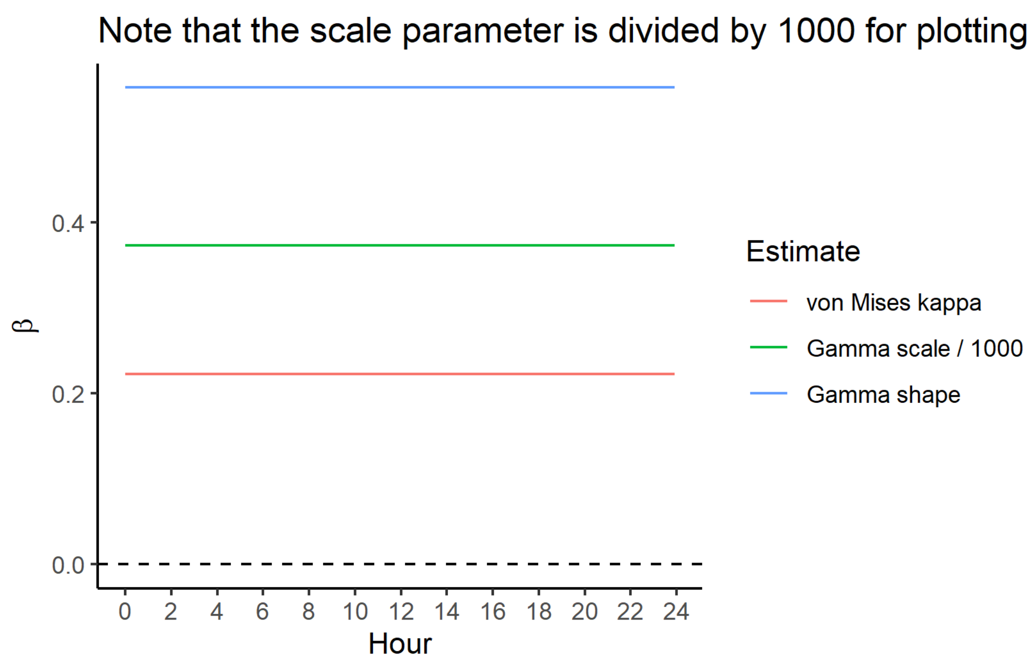
```
ggplot() +
  geom_path(data = hour_coefs_nat_long_3p %>%
              filter(!coef %in% c("shape", "scale", "kappa")),
            aes(x = hour, y = value, colour = coef)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_y_continuous(expression(Time-varying~parameter~values~beta)) +
  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

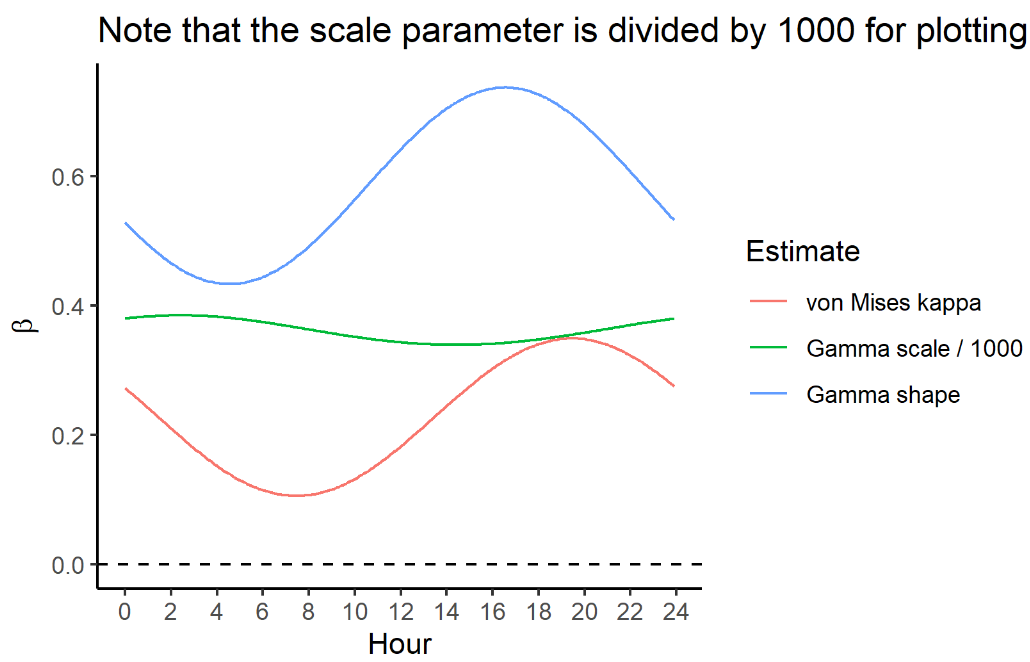
0p

```
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")
```



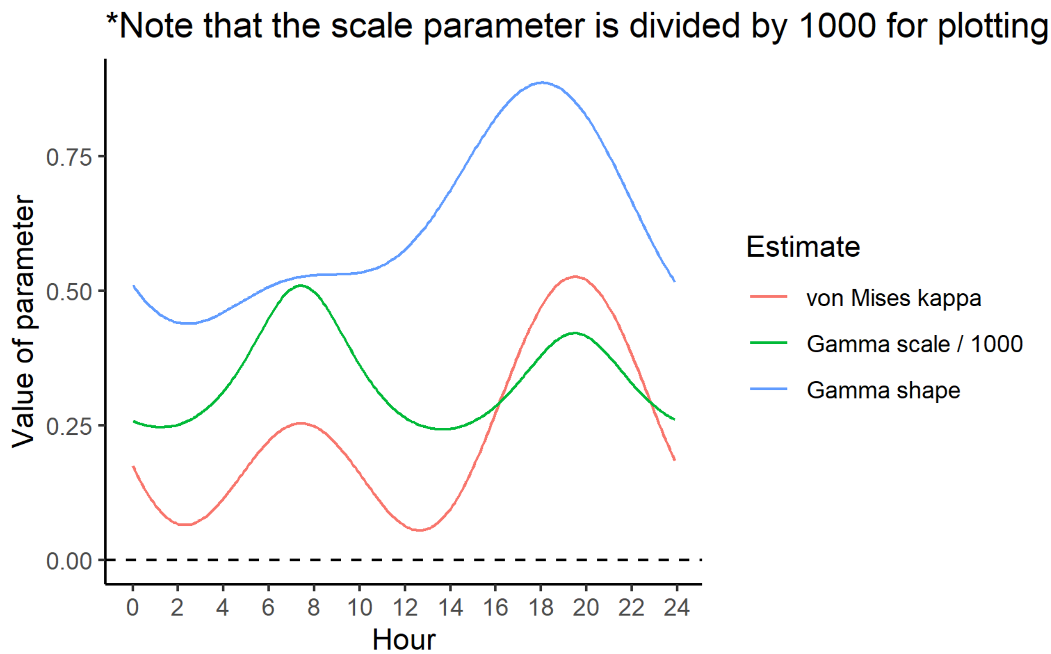
1p

```
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

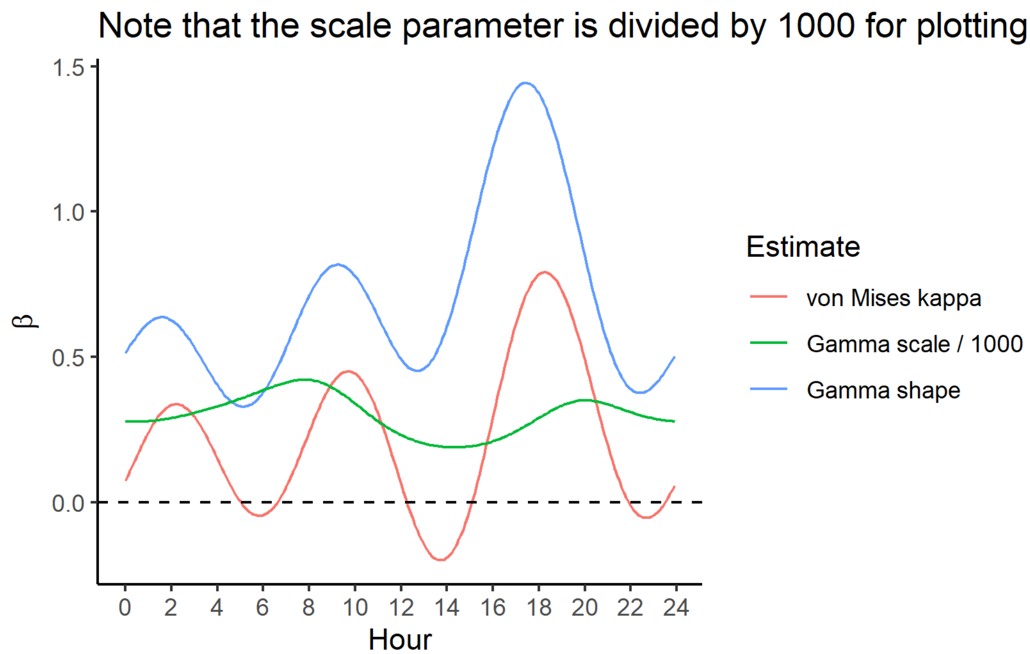
```
ggplot() +
    geom_path(data = hour_coefs_nat_long_2p %>%
                filter(coef %in% c("shape", "kappa")),
                aes(x = hour, y = value, colour = coef)) +
  geom_path(data = hour_coefs_nat_long_2p %>%
                filter(coef == "scale"),
                aes(x = hour, y = value/1000, 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.

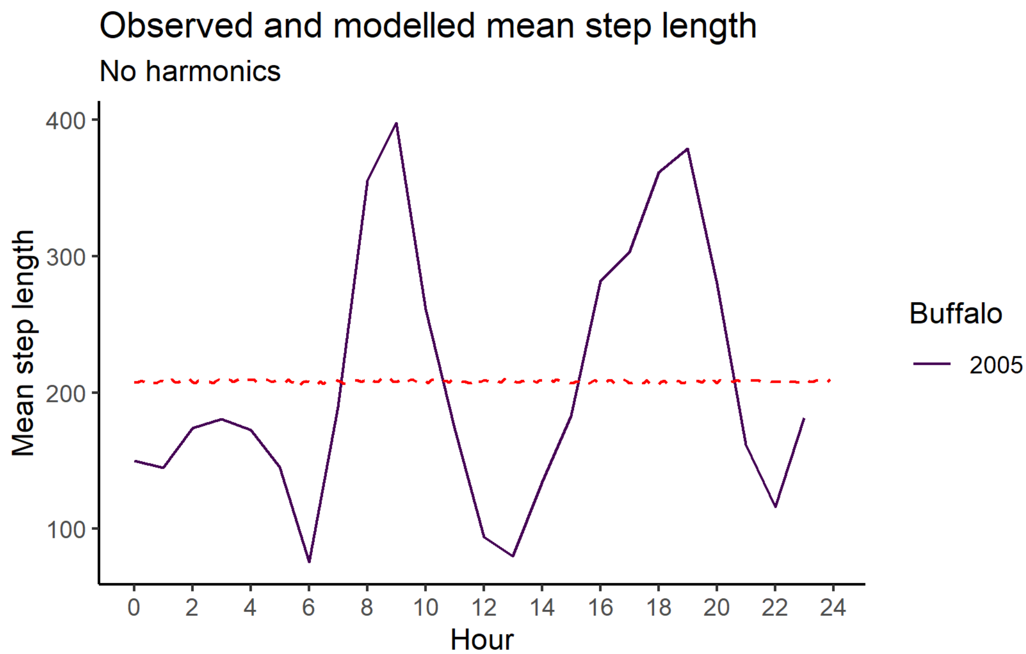
0p

```
# 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))
```

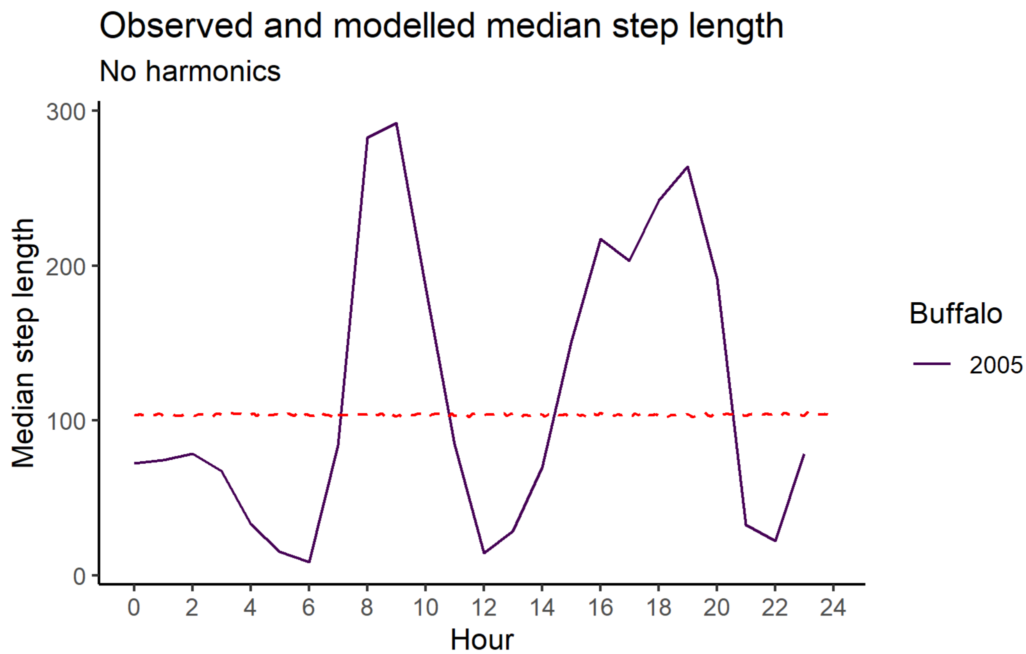
```
`summarise()` has grouped output by 'id'. You can override using the `.groups`
argument.
```

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

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




```r
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
```

```r
# 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
```

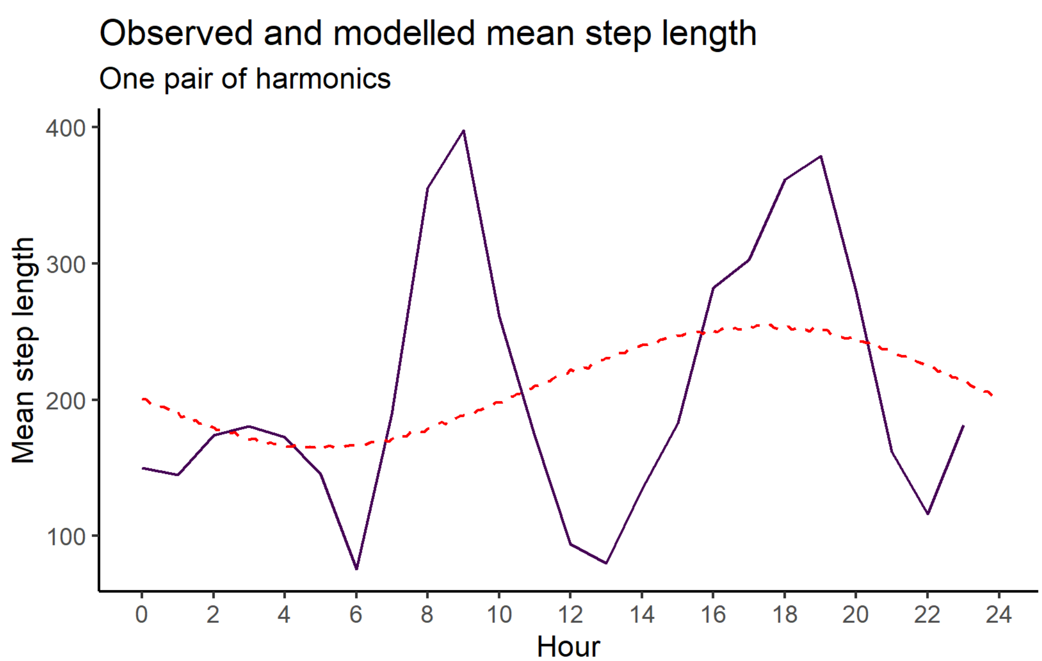
```r
# fitted model
gamma_df_0p %>% summarise(mean_mean = mean(mean),
                         median_mean = mean(median),
                         ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1    208.22    103.7539   2.006865
```

1p

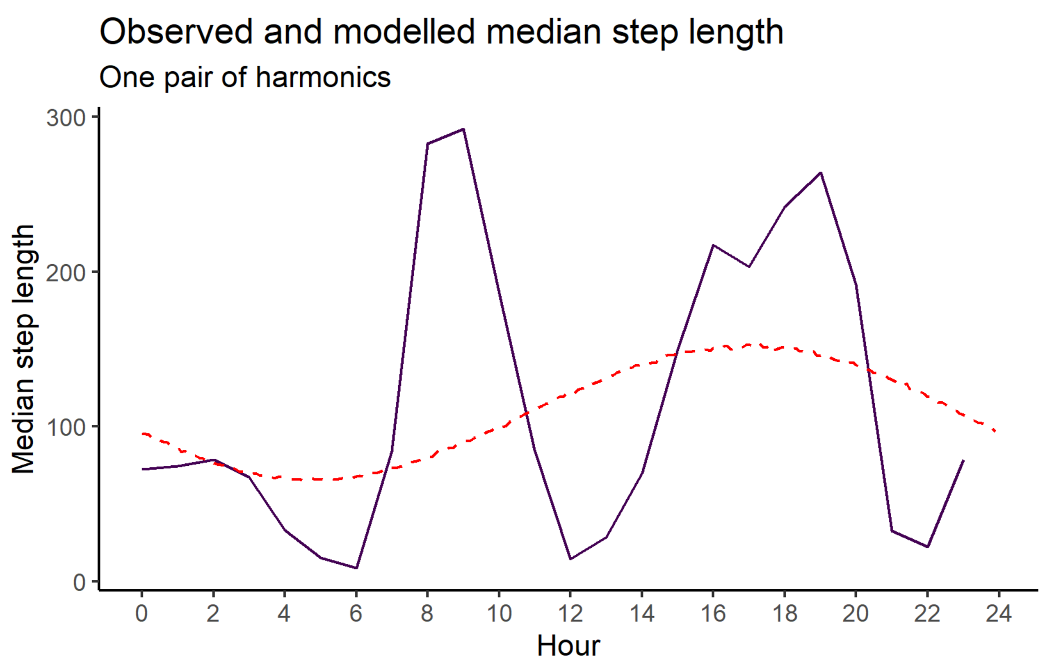
```r
gamma_dist_list <- vector(mode = "list", length = nrow(hour_coefs_nat_df_1p))
gamma_mean <- c()
gamma_median <- c()
gamma_ratio <- c()

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") +
```

```r
  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
```




```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
```

```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
```

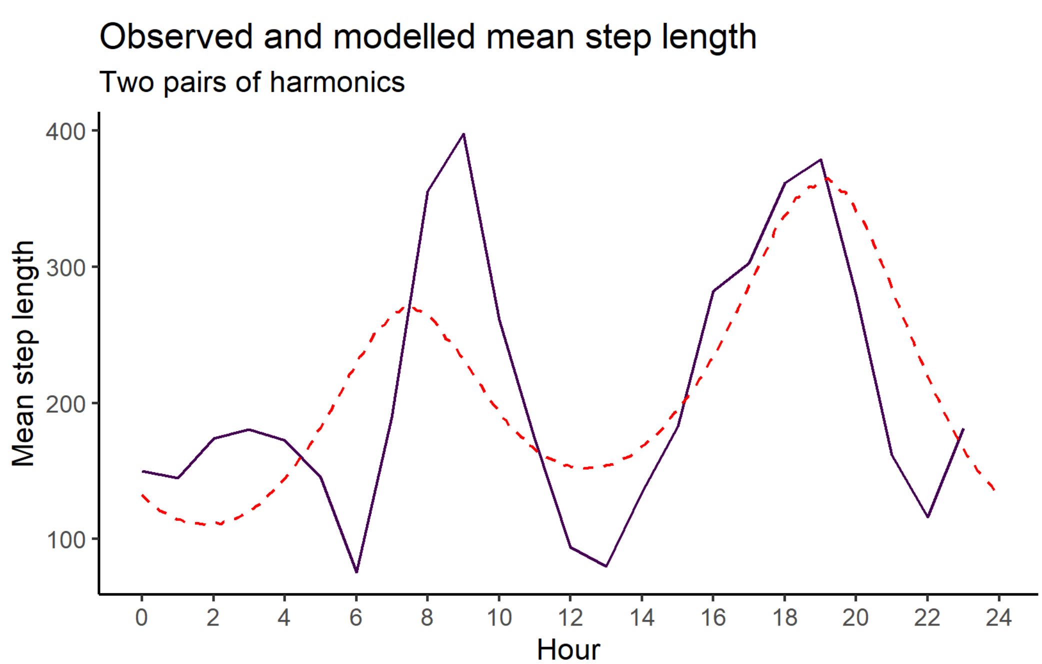
```r
gamma_df_1p %>% summarise(mean_mean = mean(mean),
                          median_mean = mean(median),
                          ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1   210.371    109.0275   1.929522
```

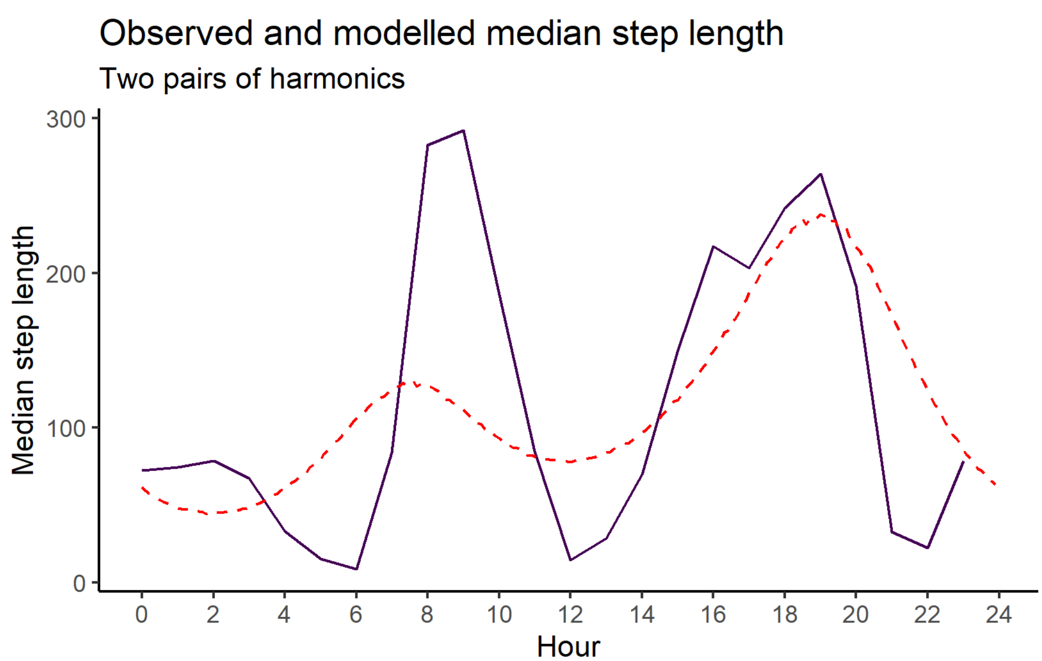
2p

```r
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]])
```

```r
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
```



```
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
```




```
# 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
```

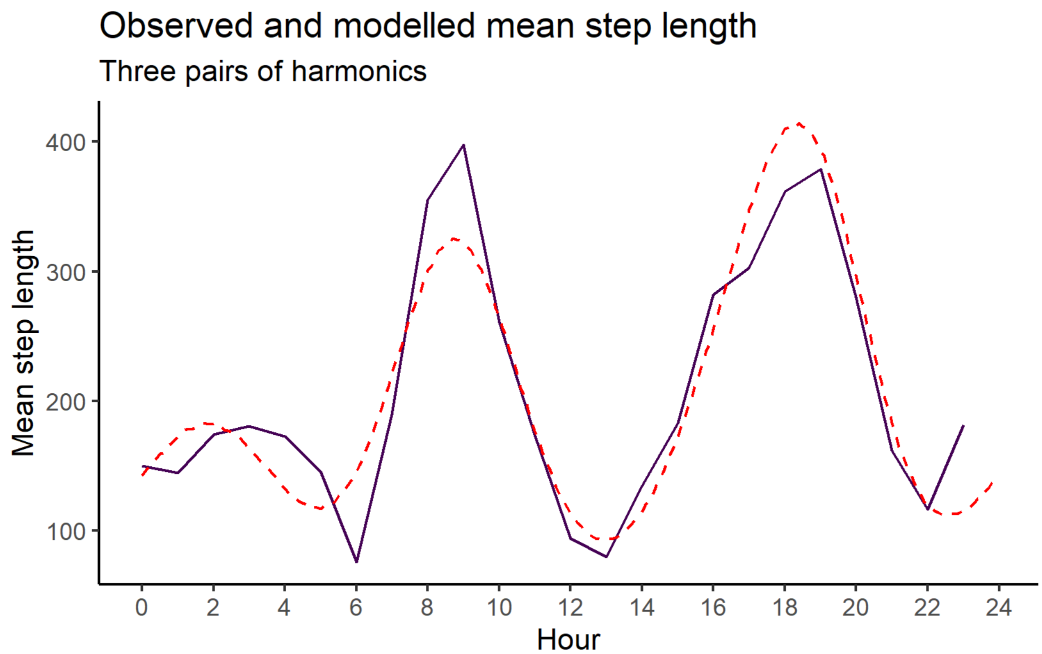
```r
gamma_df_2p %>% summarise(mean_mean = mean(mean),
                          median_mean = mean(median),
                          ratio_mean = mean_mean/median_mean)
```

```
  mean_mean median_mean ratio_mean
1  210.6744    114.9287   1.833087
```

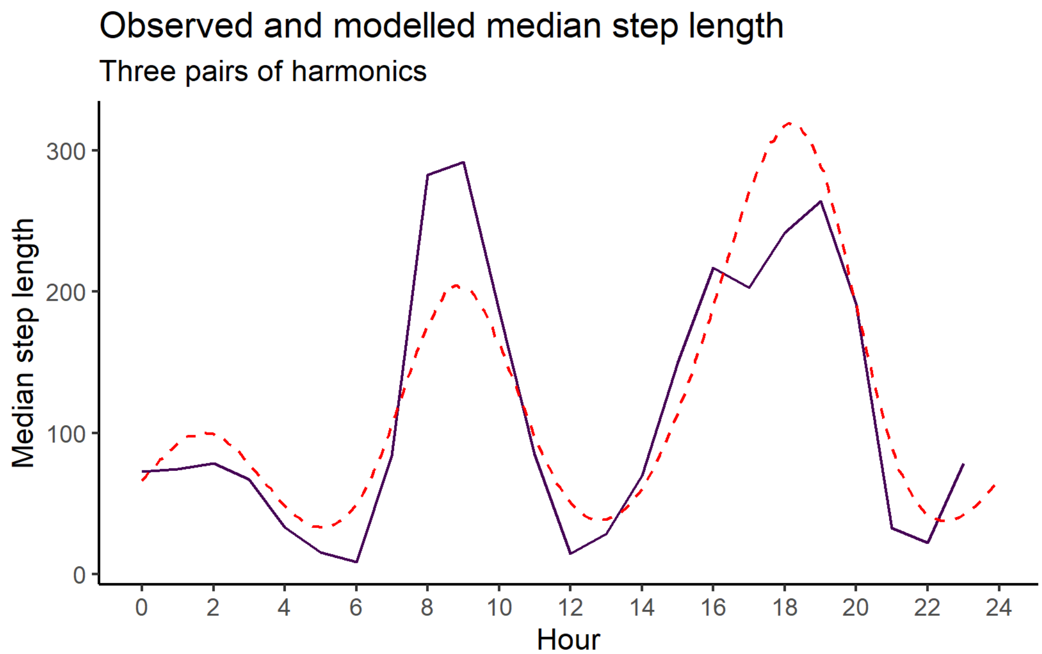
3p

```r
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)) {
```

```r
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  206.3967    121.2081   1.702829
```

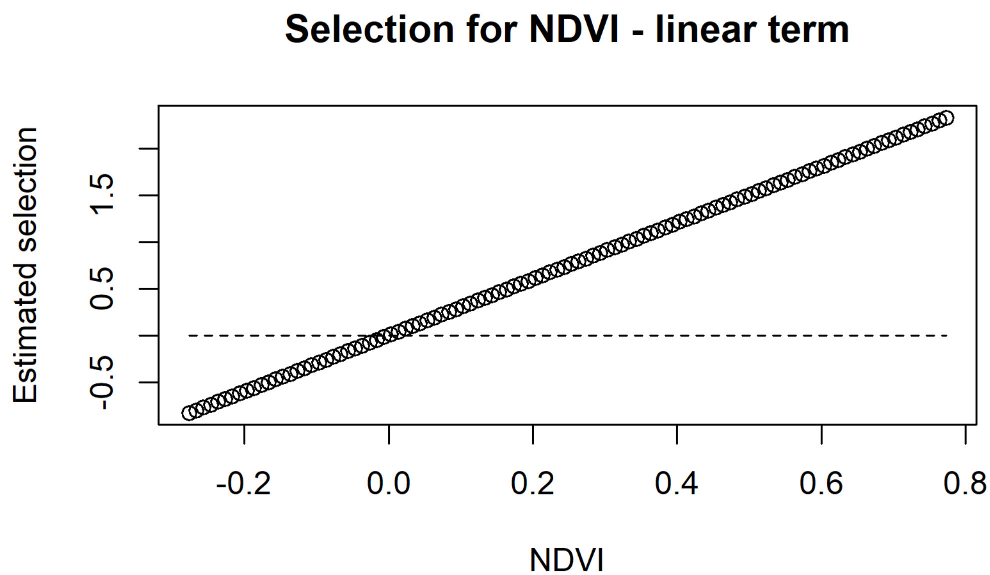
Creating selection surfaces

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.

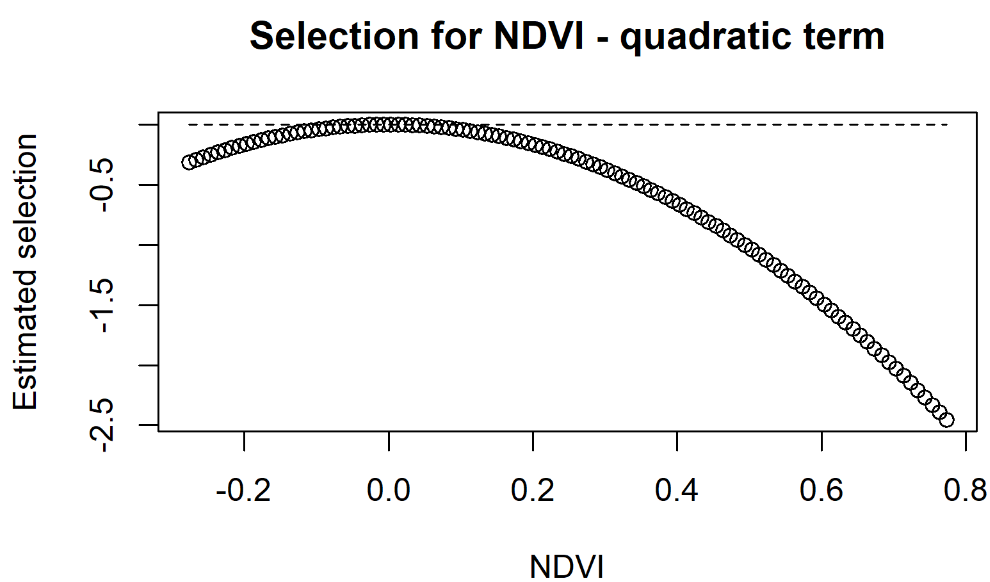
To illustrate, if we don't have temporal dynamics (as is the case for this model), then we have a coefficient for the linear term and a coefficient for the quadratic term. Using these, we can plot the selection curve at the scale of the environmental variable (in this case NDVI).

Using the natural scale coefficients from the model:

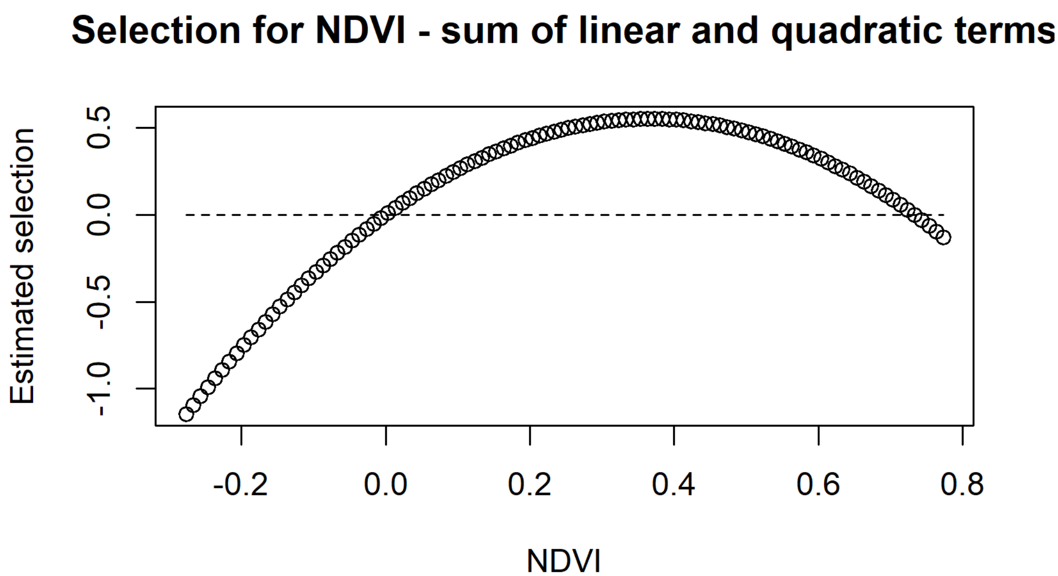
```r
# first get a sequence of NDVI values,
# starting from the minimum observed in the data to the maximum
ndvi_min <- min(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_max <- max(buffalo_data$ndvi_temporal, na.rm = TRUE)
ndvi_seq <- seq(ndvi_min, ndvi_max, by = 0.01)

# 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")
```



```r
# 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.

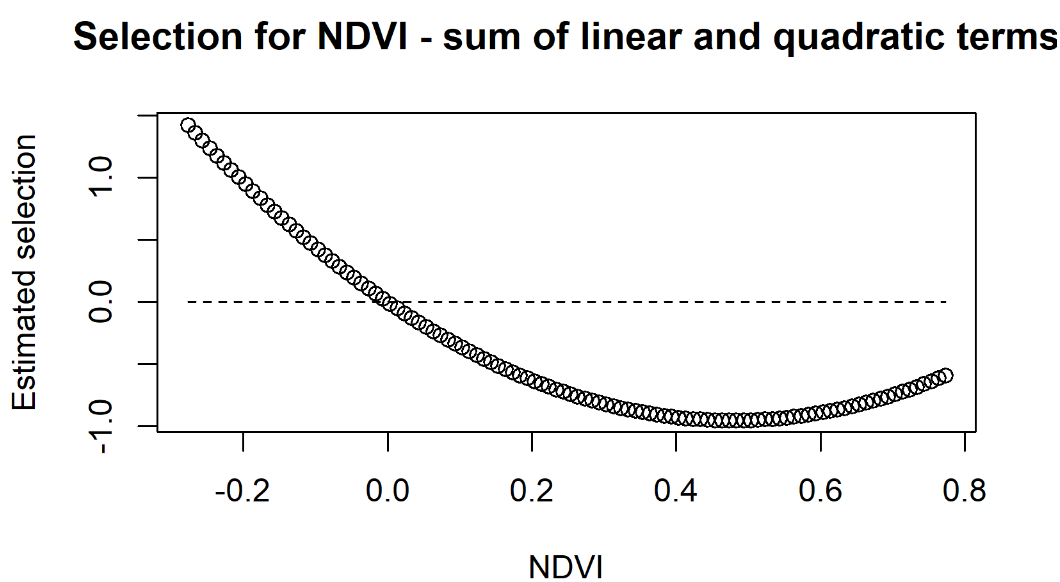
Here we illustrate for the model with 2 pairs of harmonic terms.

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

First for **Hour 3**

```r
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")
```

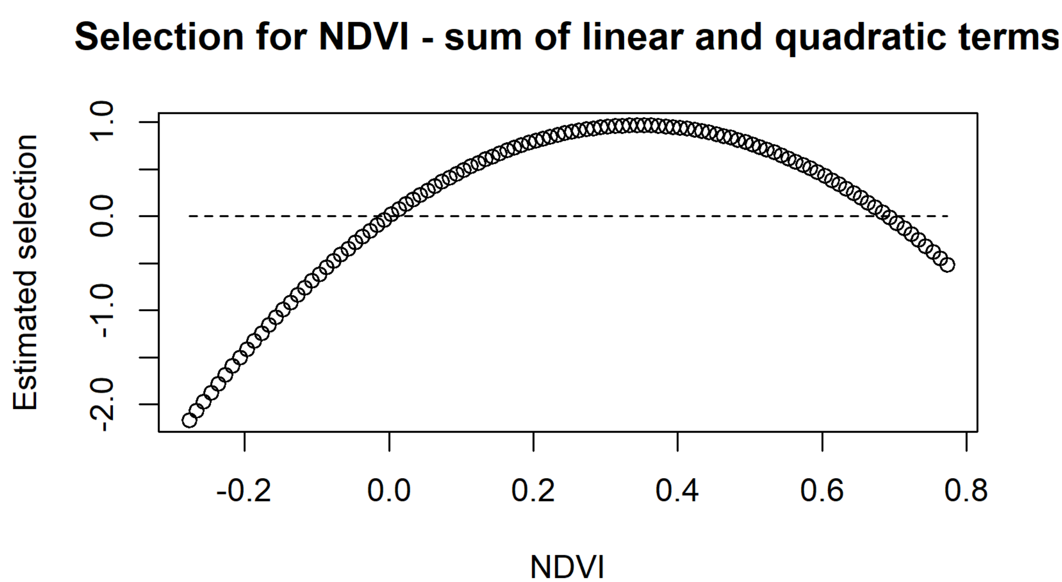


We can see that the coefficient at hour 3 shows highest selection for NDVI values slightly above 0.2, and the coefficient is mostly negative.

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.

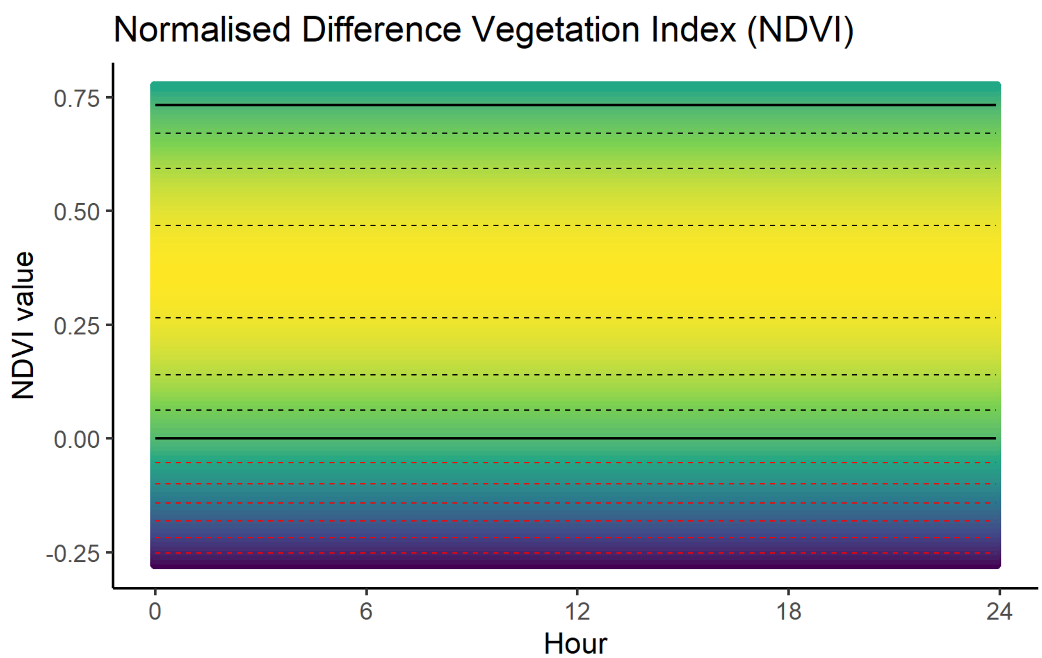
We simply index over the linear and quadratic terms and calculate the coefficient values at every time point.

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),
```

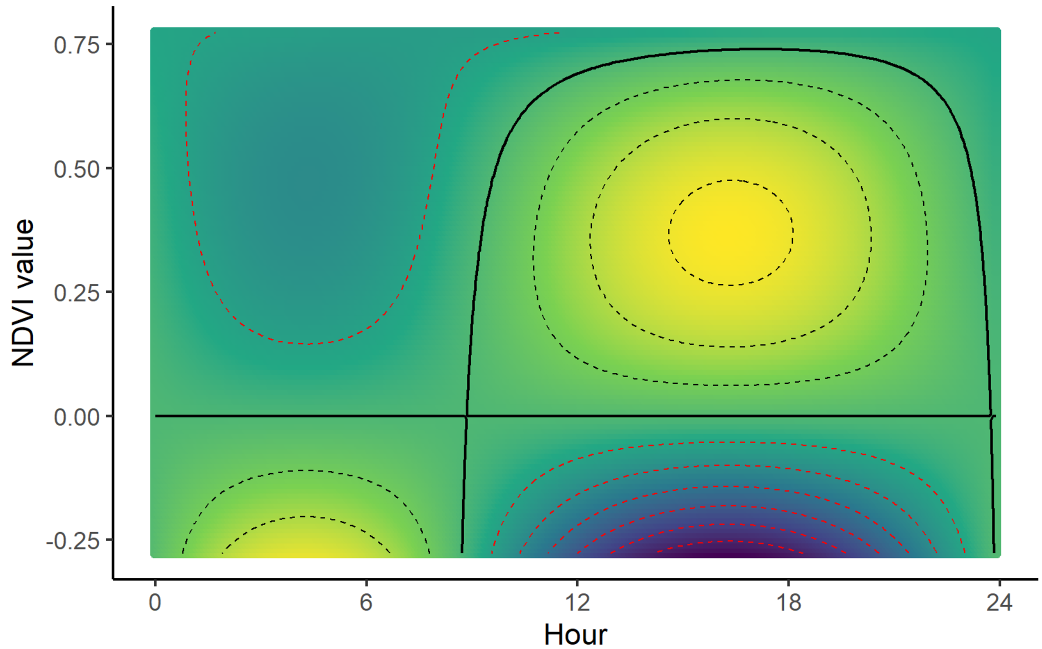
```r
                    colour = "red", linewidth = 0.25, linetype = "dashed") +
  geom_contour(aes(z = value), breaks = 0, colour = "black", linewidth = 0.5) +
  scale_x_continuous("Hour", breaks = seq(0,24,6)) +
  scale_y_continuous("NDVI value", breaks = seq(-1, 1, 0.25)) +
  scale_colour_viridis_c("Selection") +
  ggtitle("Normalised Difference Vegetation Index (NDVI)") +
  theme_classic() +
  theme(legend.position = "none")

ndvi_quad_0p
```



1p

```r
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
```

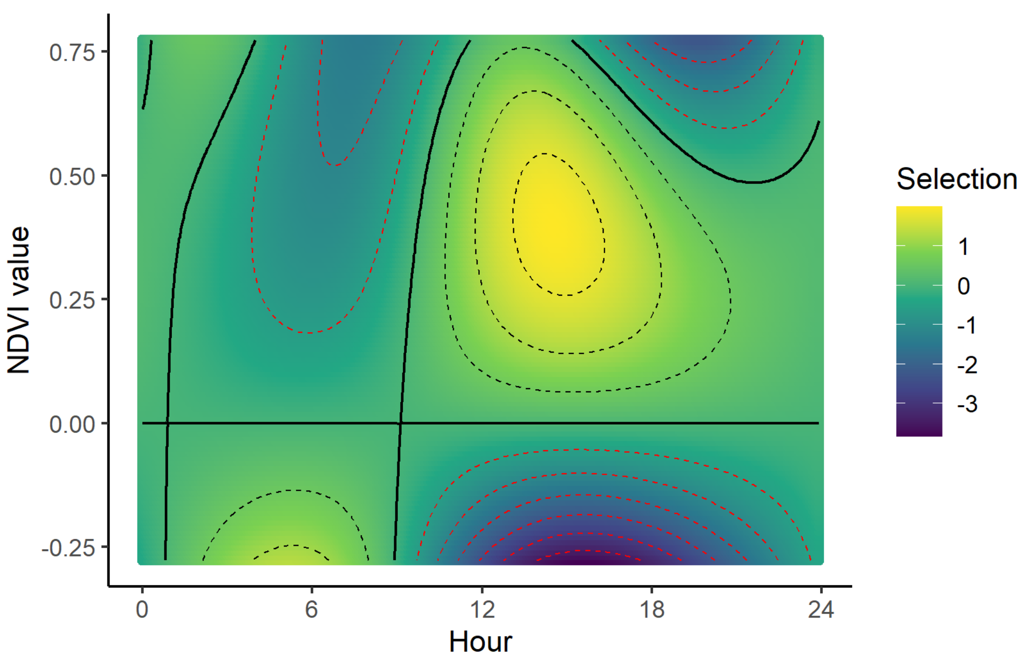
```r
  ndvi_fresponse_df[,i] <- (hour_coefs_nat_df_1p$ndvi[i] * ndvi_seq) +
    (hour_coefs_nat_df_1p$ndvi_2[i] * (ndvi_seq ^ 2))
}

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
```

```r
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
```

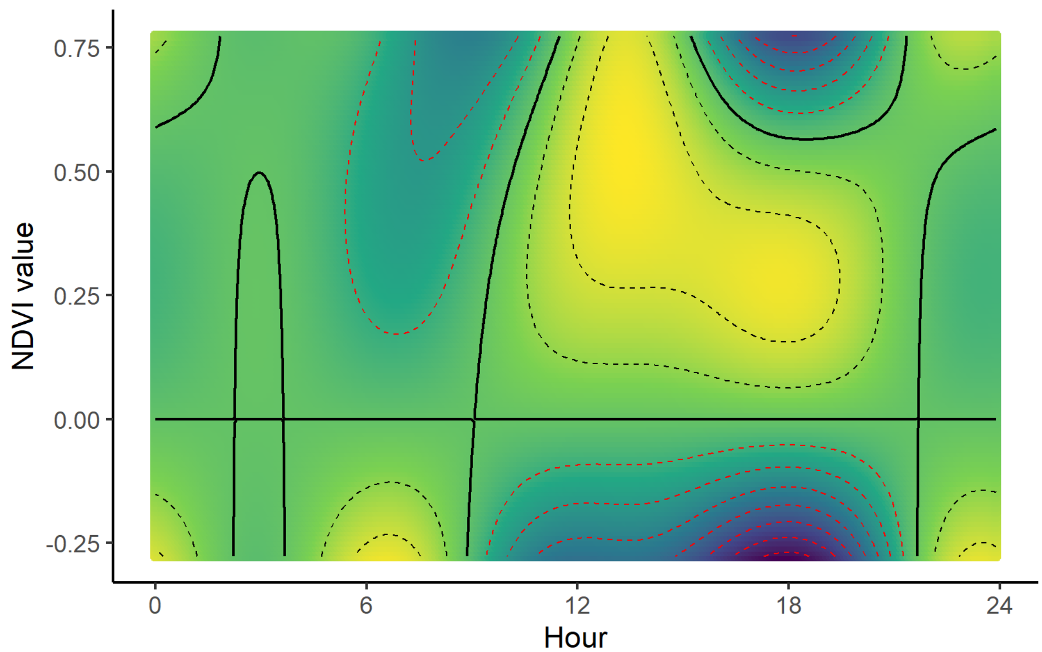


```
# ggsave(paste0("outputs/plots/manuscript_figs_R2/ndvi_selection_surface_legend_",
#          Sys.Date(), ".png"),
#   width=170, height=90, units="mm", dpi = 1000)
```

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))
}

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) +
```

```r
  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
```

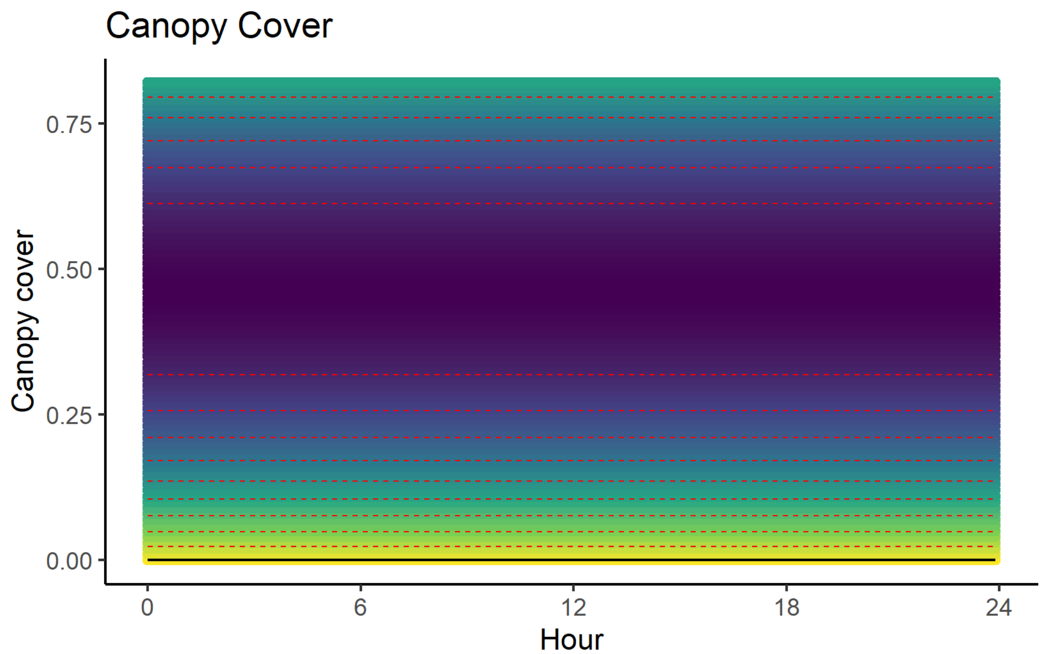


Canopy cover selection surface

0p

```r
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
```

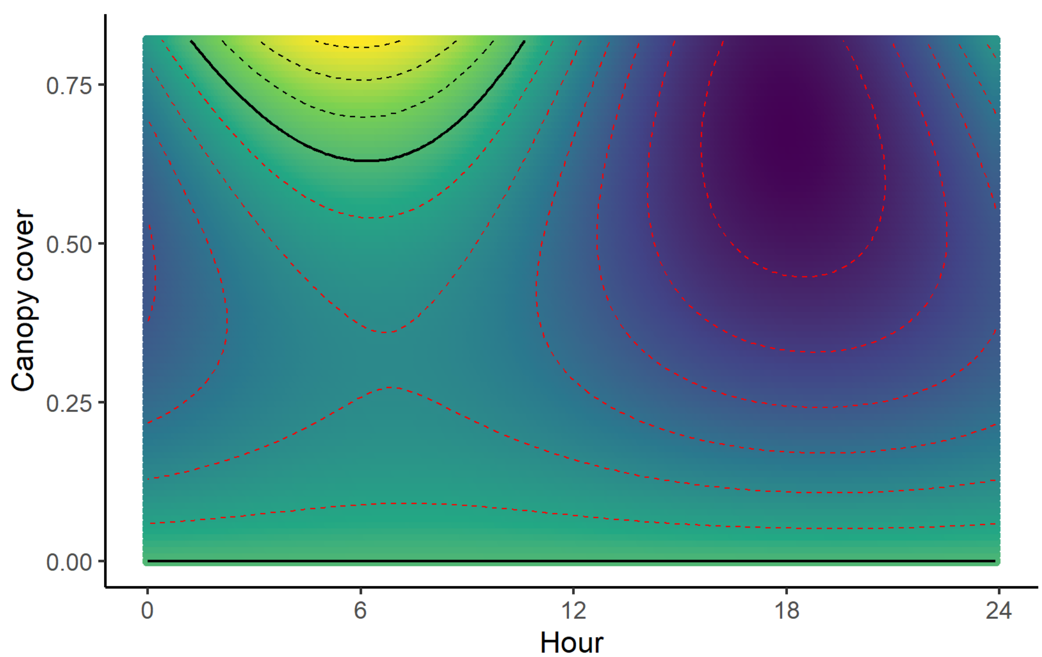
```
  canopy_fresponse_df[,i] <- (hour_coefs_nat_df_0p$canopy[i] * canopy_seq) +
    (hour_coefs_nat_df_0p$canopy_2[i] * (canopy_seq ^ 2))
}

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$value) # 0
canopy_contour_max <- max(canopy_fresponse_long$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
```

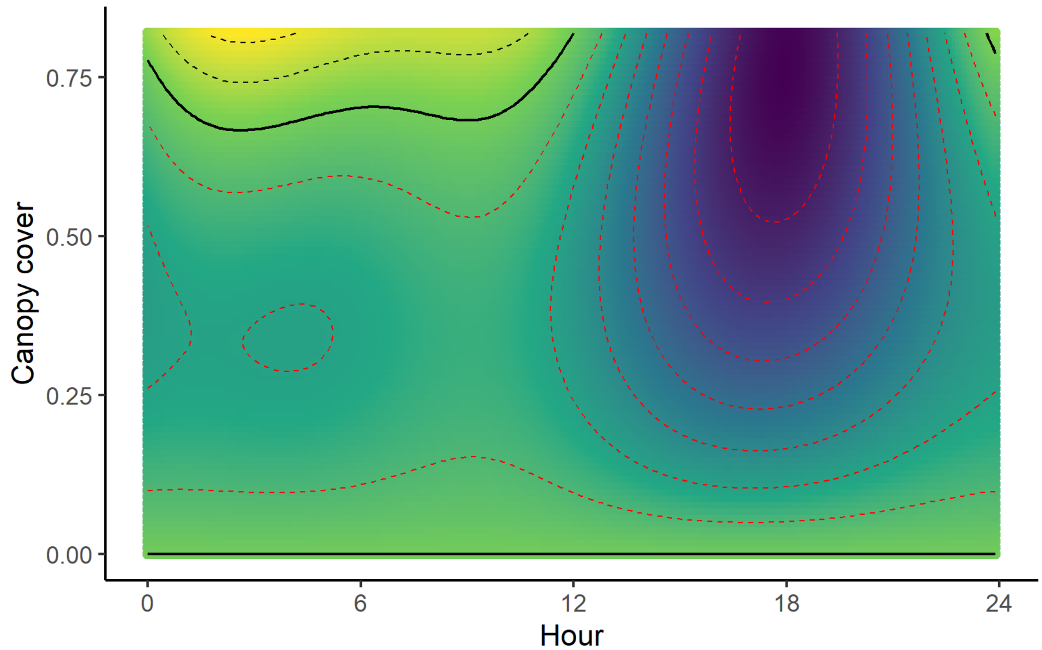
```r
canopy_quad_1p <- ggplot(data = canopy_fresponse_long,
                         aes(x = as.numeric(hour), y = canopy)) +
  geom_point(aes(colour = value)) +
  geom_contour(aes(z = value),
               breaks = seq(canopy_contour_increment,
                            canopy_contour_max,
                            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
```



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$canopy[i] * canopy_seq) +
    (hour_coefs_nat_df_2p$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() +
```

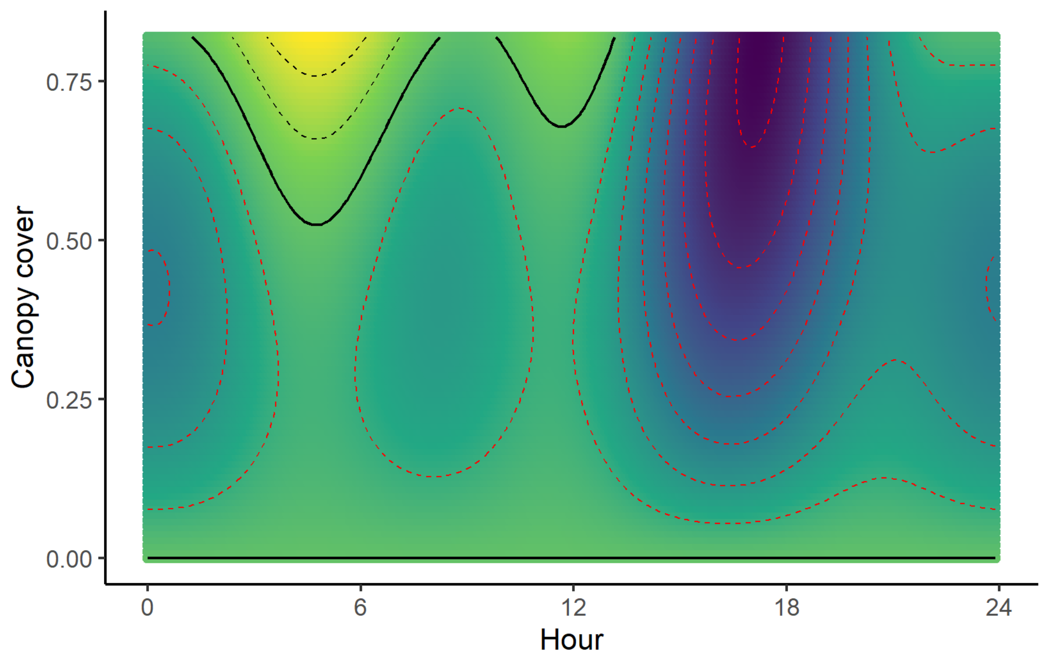
```
  theme(legend.position = "none")

canopy_quad_2p
```



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,
```

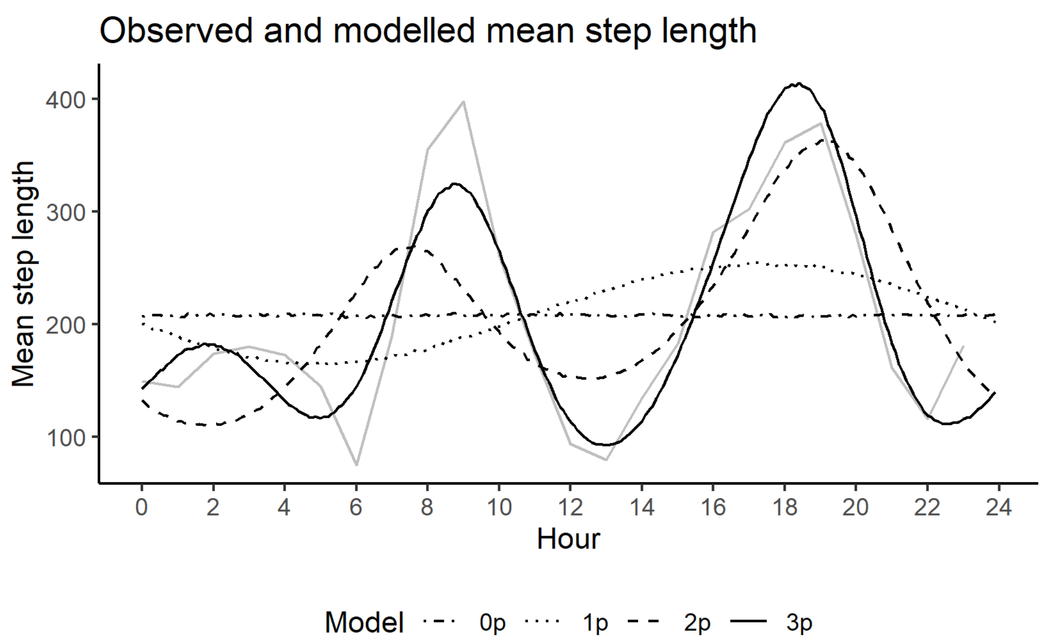
```
                                      names_to = "hour")

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

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
```



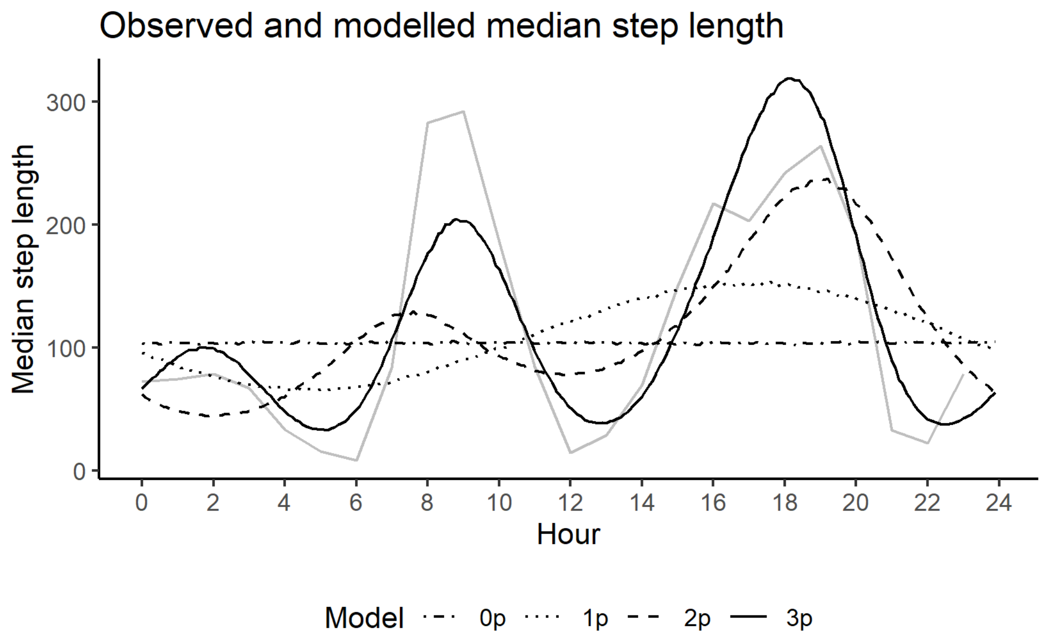
Combining the plots

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
```

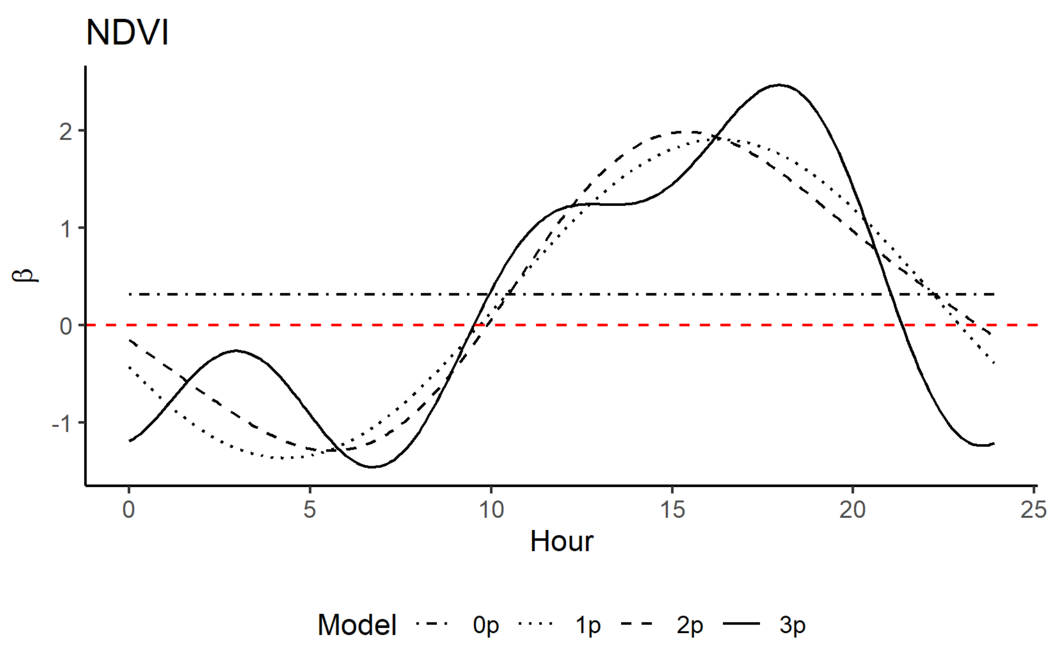



```
# ggsave(paste0("outputs/plots/manuscript_figs_R1/median_sl_",
#          Sys.Date(), ".png"),
#   width=150, height=90, units="mm", dpi = 1000)
```

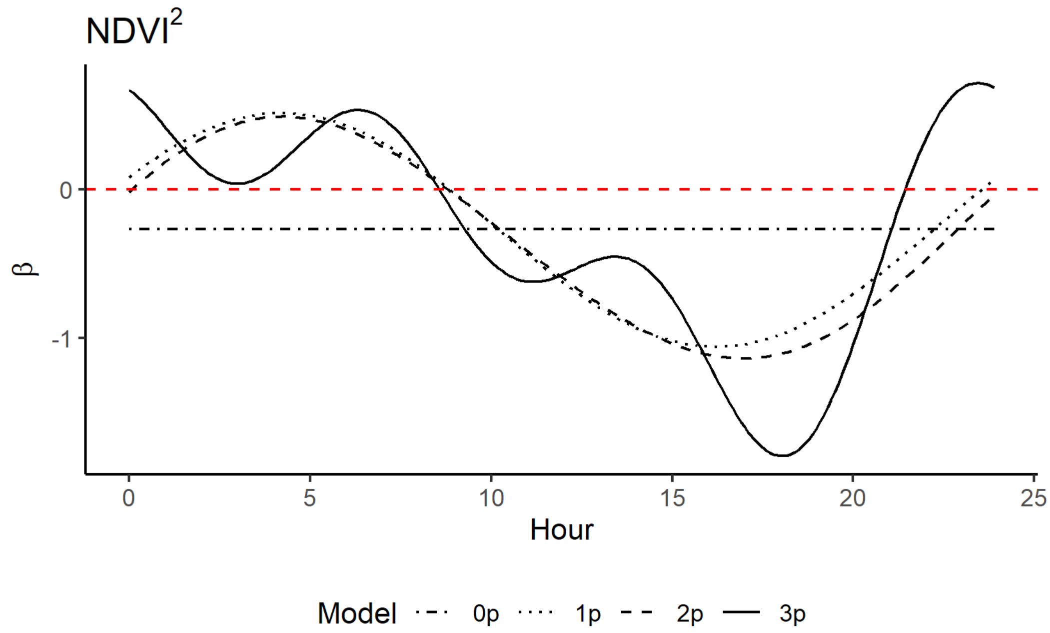
Habitat selection

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

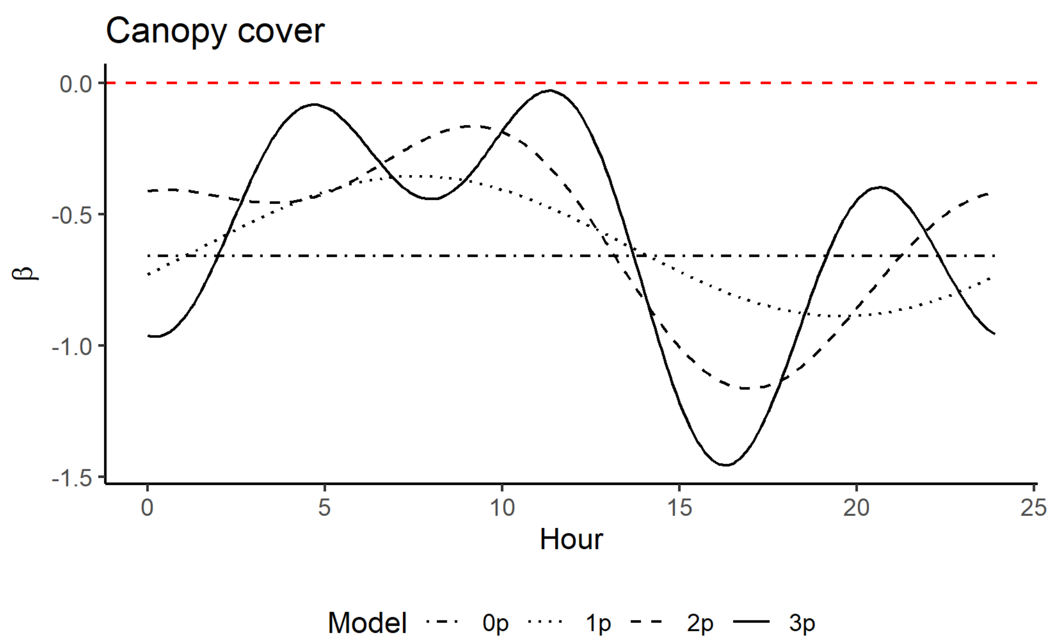
```
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      scale_x_continuous("Hour") +
  scale_linetype_manual("Model", breaks=c("0p","1p", "2p", "3p"),
                        values=c(4,3,2,1)) +
      ggtitle("NDVI") +
      theme_classic() +
      theme(legend.position = "bottom")

ndvi_harms
```



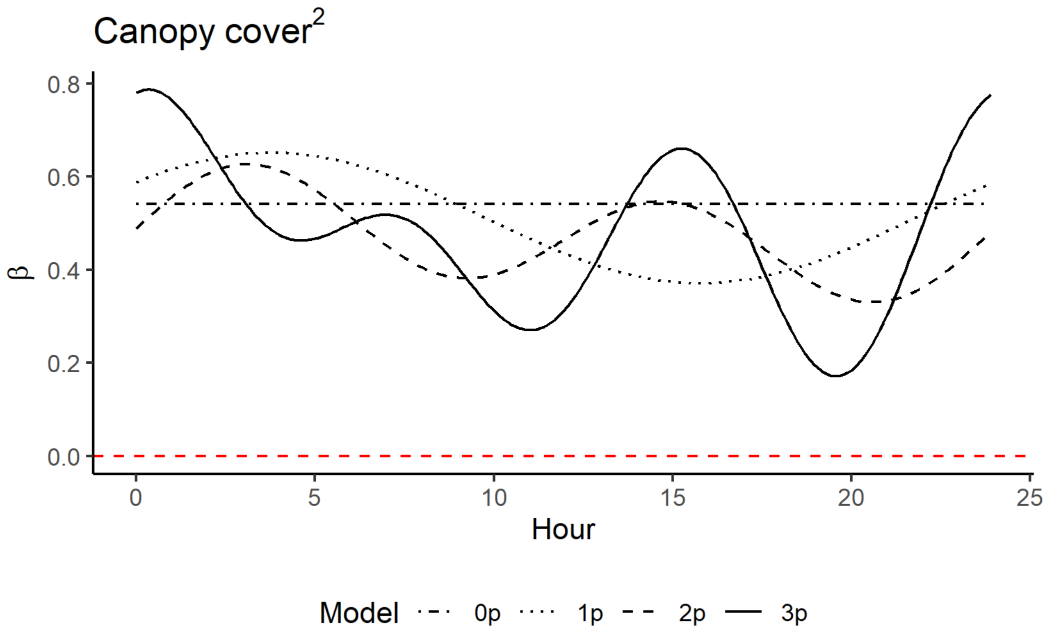

```
ndvi_2_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "ndvi_2"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      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
```



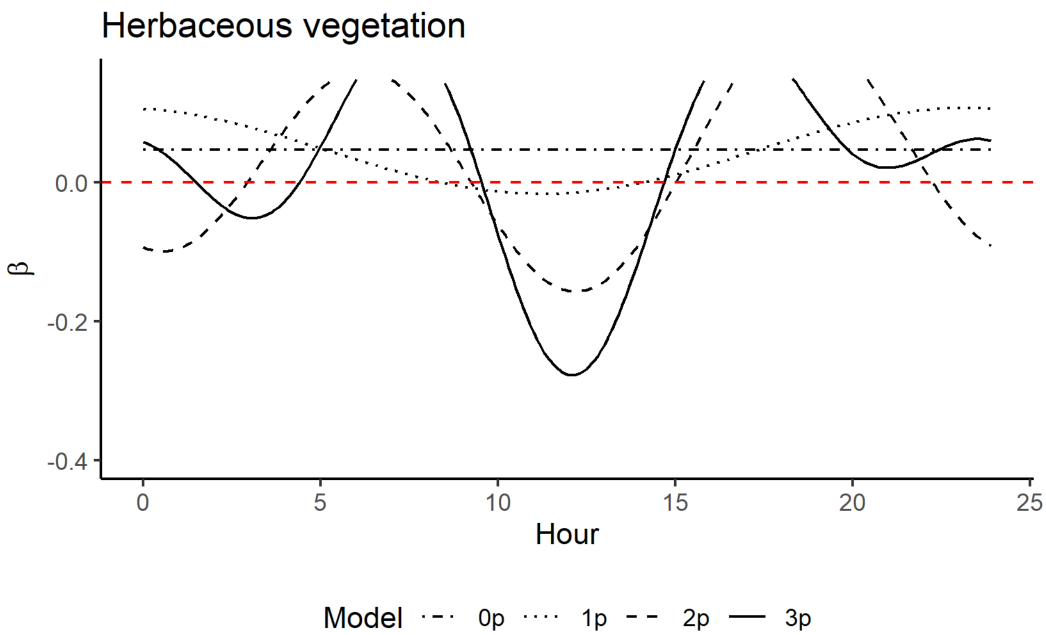

```
canopy_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "canopy"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      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
```



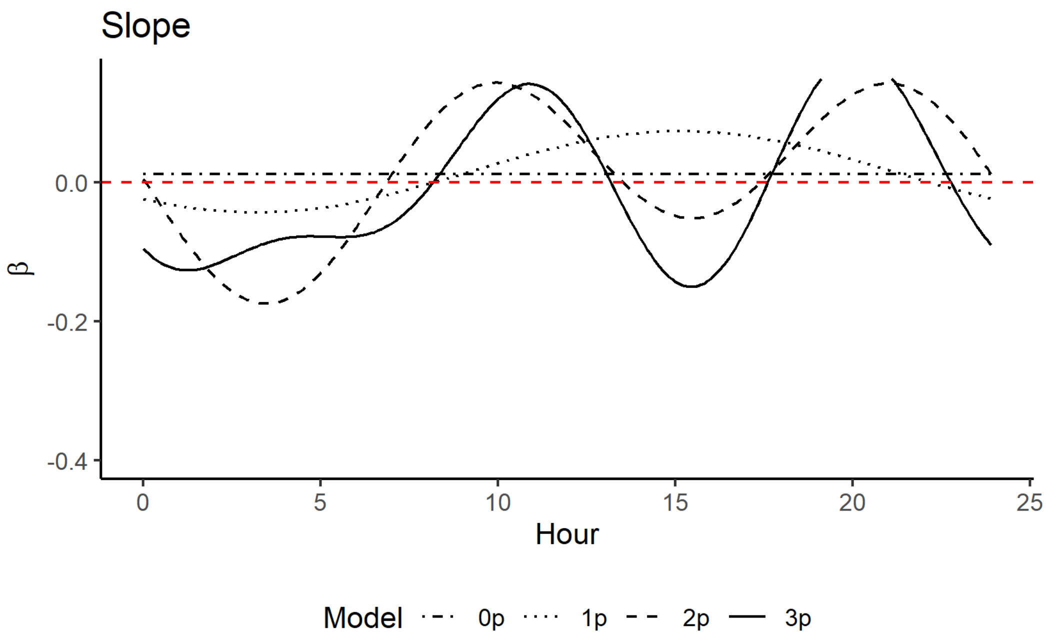

```r
canopy_2_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "canopy_2"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta)) +
      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
```



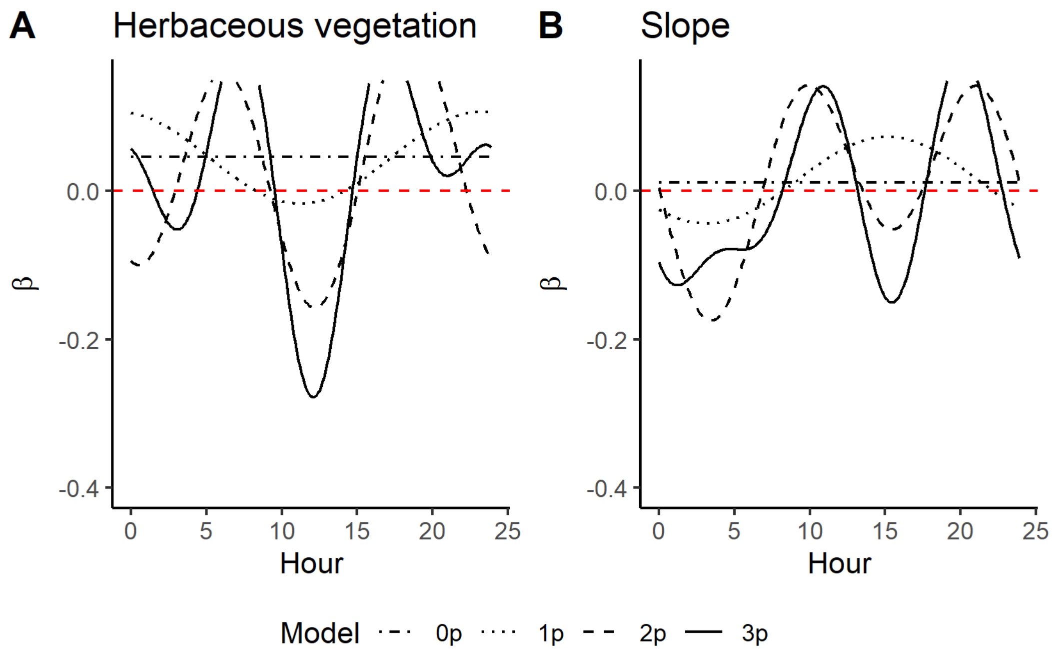

```
herby_harms <- ggplot() +
      geom_path(data = harmonics_scaled_long_Mp %>%
                filter(coef == "herby"),
                aes(x = hour, y = value, linetype = model)) +
      geom_hline(yintercept = 0, linetype = "dashed", colour = "red") +
      scale_y_continuous(expression(beta), limits = 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
```




```
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)
```

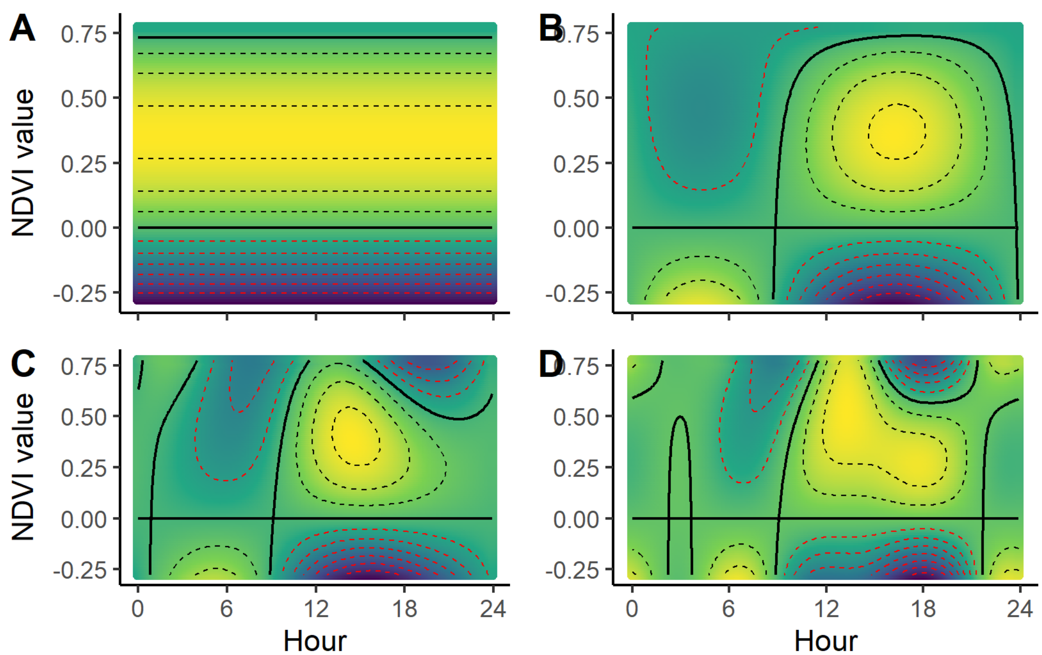
Combining selection surfaces

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(),
                               ),
```

```r
          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)
```

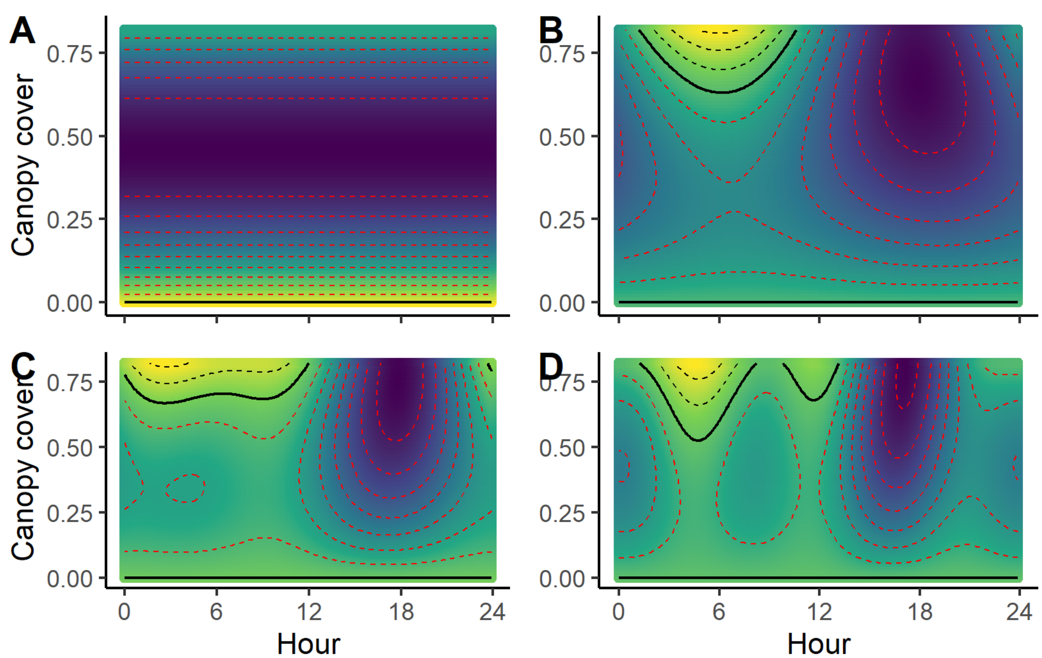



```r
# ggsave(paste0("outputs/plots/manuscript_figs_R1/",
#                "NDVI_2x2_CLR_TS_daily_GvM_10rs_",
#           Sys.Date(), ".png"),
#   width=150, height=120, units="mm", dpi = 1000)
```

Canopy cover

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

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

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




```r
# 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

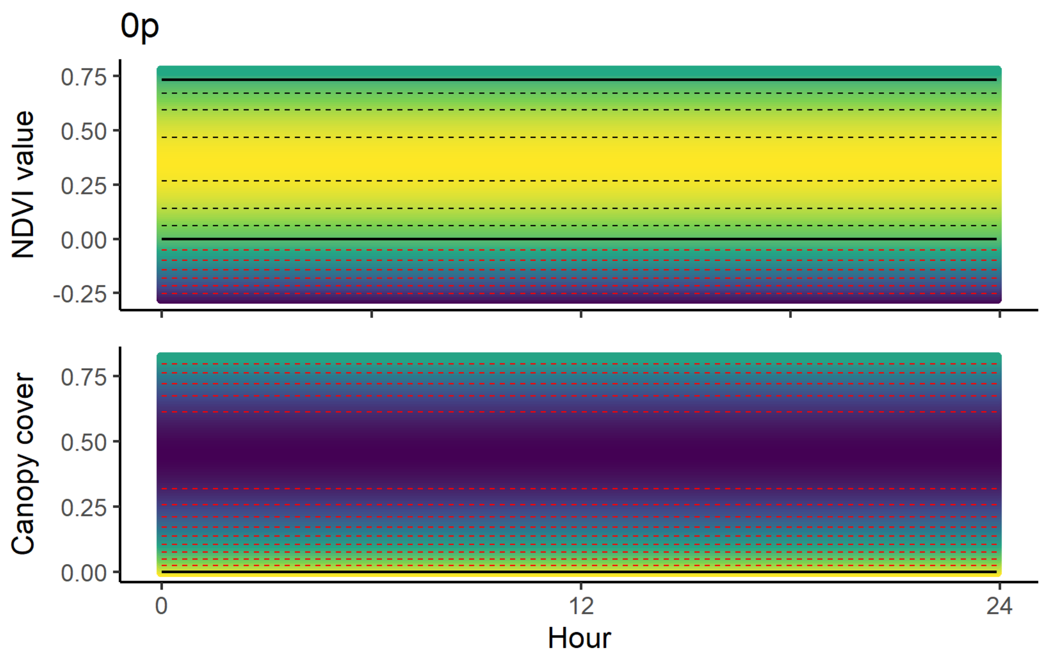
We combine these plots into the plot that is in the paper. On the top is the **NDVI** selection surface, and on the bottom is the **canopy cover** selection surface.

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)
```

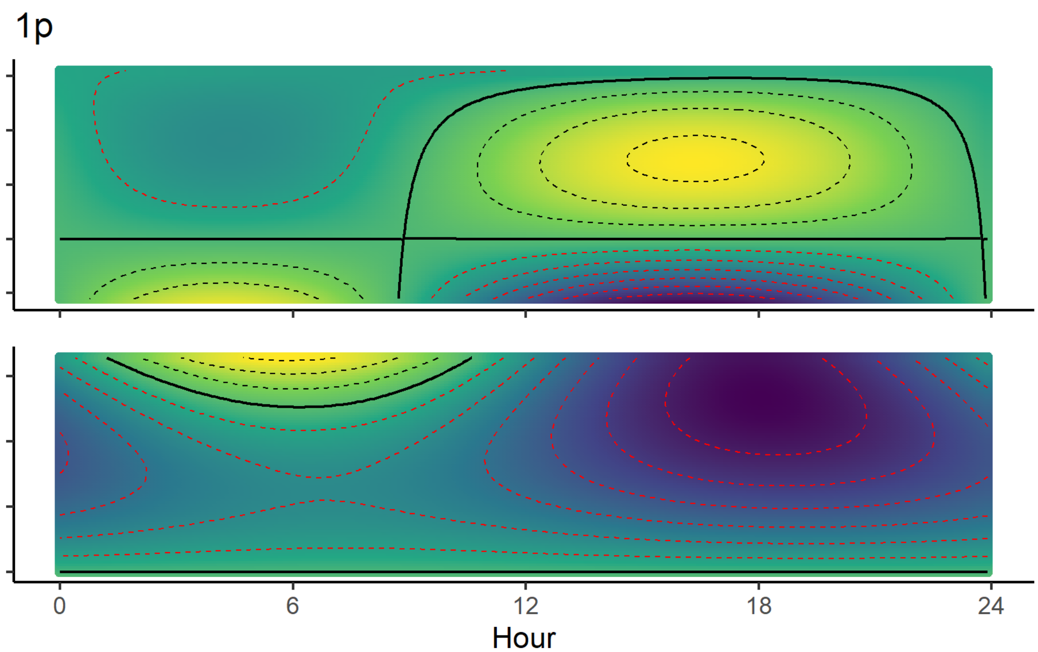
```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

```
surface_plots_0p
```



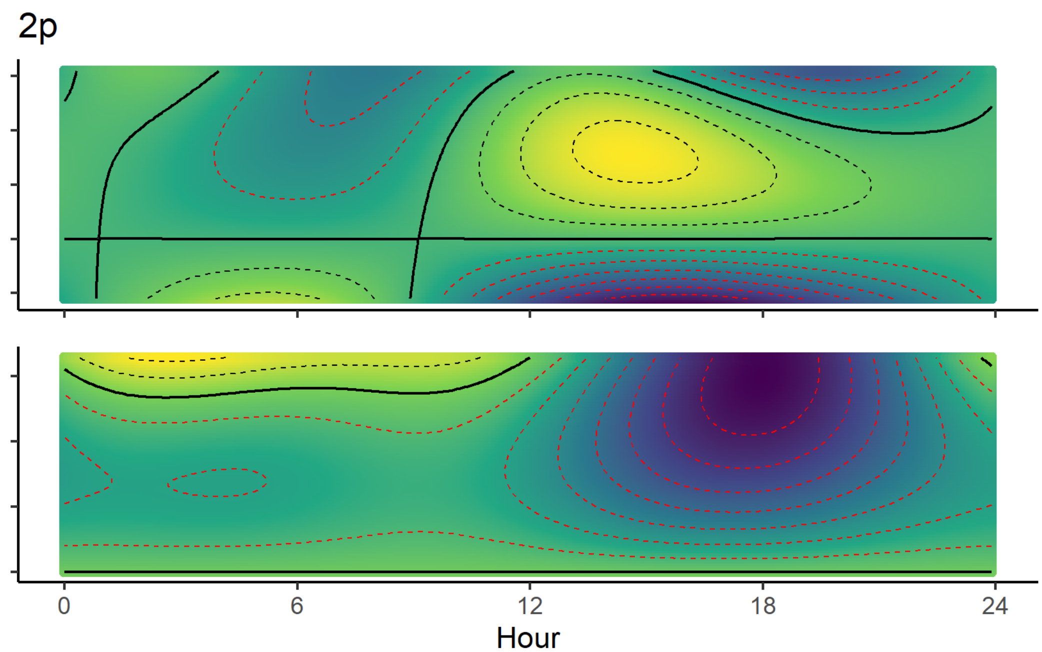
1p

```r
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)

surface_plots_1p
```



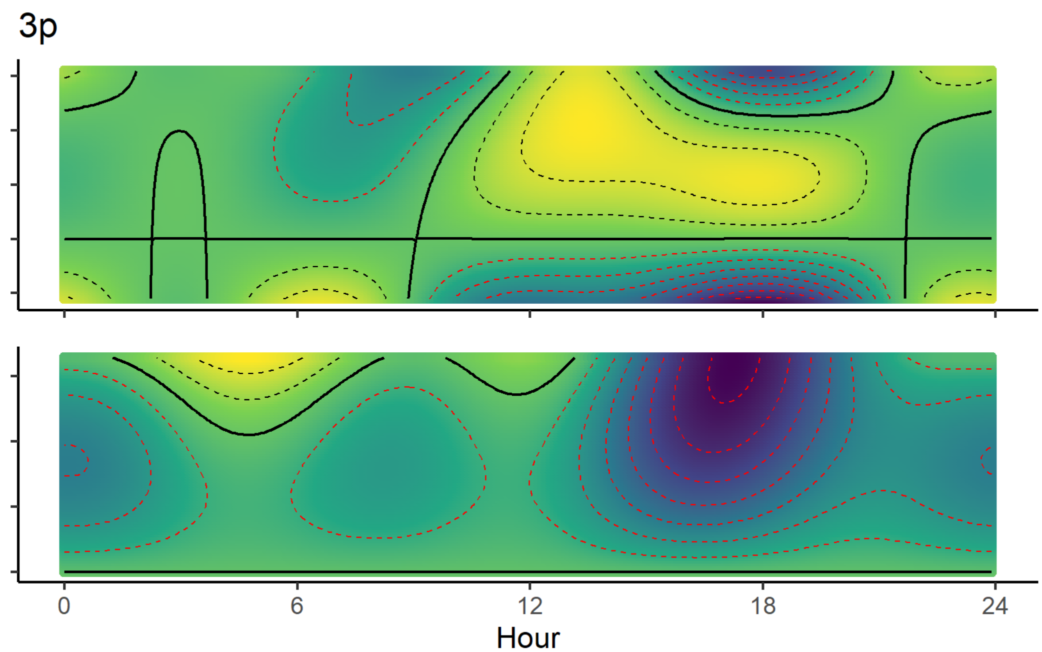
2p

```
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)

surface_plots_2p
```



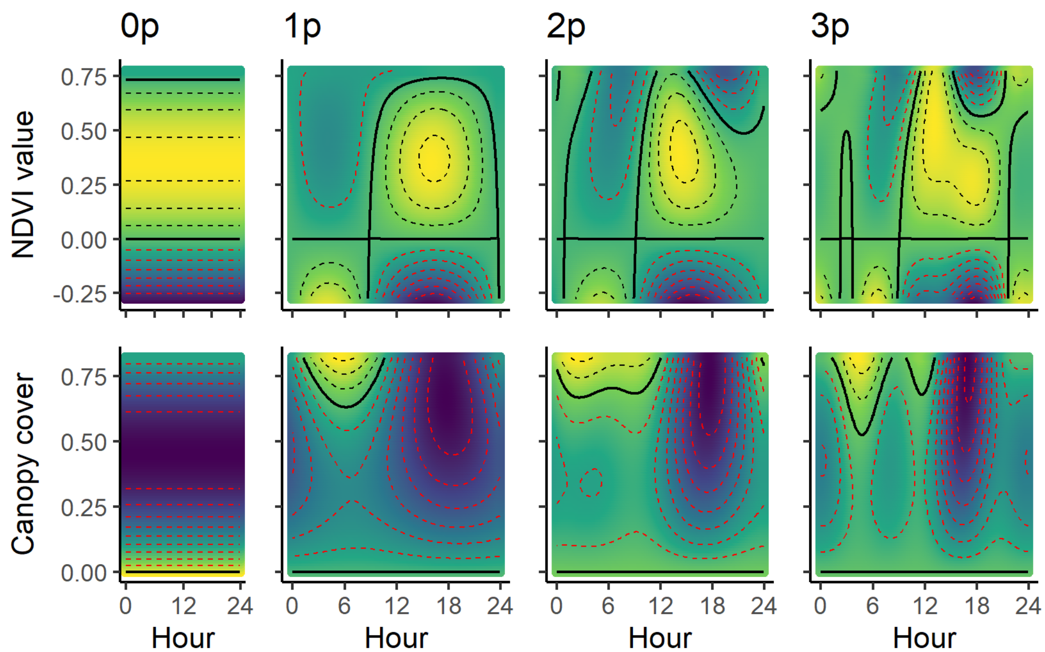
3p

```
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] scales_1.3.0    patchwork_1.3.0 MASS_7.3-60.2   ggpubr_0.6.0
 [5] beepr_2.0       tictoc_1.2.1    terra_1.7-78    survival_3.6-4
 [9] amt_0.2.2.0     lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
[13] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
[17] 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    abind_1.4-8        tidyselect_1.2.1   digest_0.6.37
[33] stringi_1.8.4      sf_1.0-17          labeling_0.4.3     splines_4.4.1
[37] cowplot_1.1.3      fastmap_1.2.0      grid_4.4.1         colorspace_2.1-1
[41] cli_3.6.3          magrittr_2.0.3     utf8_1.2.4         broom_1.0.6
[45] e1071_1.7-16       withr_3.0.1        backports_1.5.0    bit64_4.0.5
[49] timechange_0.3.0   rmarkdown_2.28     audio_0.1-11       bit_4.0.5
[53] gridExtra_2.3      ggsignif_0.6.4     hms_1.1.3          evaluate_1.0.0
[57] knitr_1.48         rbibutils_2.2.16   viridisLite_0.4.2  rlang_1.1.4
[61] isoband_0.2.7      Rcpp_1.0.13        glue_1.7.0         DBI_1.2.3
[65] vroom_1.6.5        jsonlite_1.8.8     R6_2.5.1           units_0.8-5
```