SSF responses

Scott Forrest

2025-08-22

In this script we demonstrate different approaches to including covariates in an SSF model.

Table of contents

Load required packages

```
library(tidyverse)
packages <- c("amt", "sf", "terra", "RColorBrewer", "mgcv", "gratia")
walk(packages, require, character.only = T)
```

Import data and clean

```
buffalo_data <- read_csv("data/buffalo.csv")
```

```
New names:
Rows: 133161 Columns: 11
-- Column specification
-------------------------------------------------------- Delimiter: "," chr
(2): node, dates dbl (7): ...1, lat, lon, height, accuracy, heading, speed dttm
(2): timestamp, DateTime
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.
* `` -> `...1`
```

```
# remove individuals that have poor data quality or less than about 3 months of data.
# The "2014.GPS_COMPACT copy.csv" string is a duplicate of ID 2024, so we exclude it
buffalo_data <- buffalo_data %>% filter(!node %in% c("2014.GPS_COMPACT copy.csv",
                                          2029, 2043, 2265, 2284, 2346))

buffalo_data <- buffalo_data %>%
  group_by(node) %>%
  arrange(DateTime, .by_group = T) %>%
  distinct(DateTime, .keep_all = T) %>%
  arrange(node) %>%
  mutate(ID = node)

buffalo_clean <- buffalo_data[, c(12, 2, 4, 3)]
colnames(buffalo_clean) <- c("id", "time", "lon", "lat")
attr(buffalo_clean$time, "tzone") <- "Australia/Queensland"
head(buffalo_clean)
```

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

```
tz(buffalo_clean$time)
```

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

```
buffalo_ids <- unique(buffalo_clean$id)
```

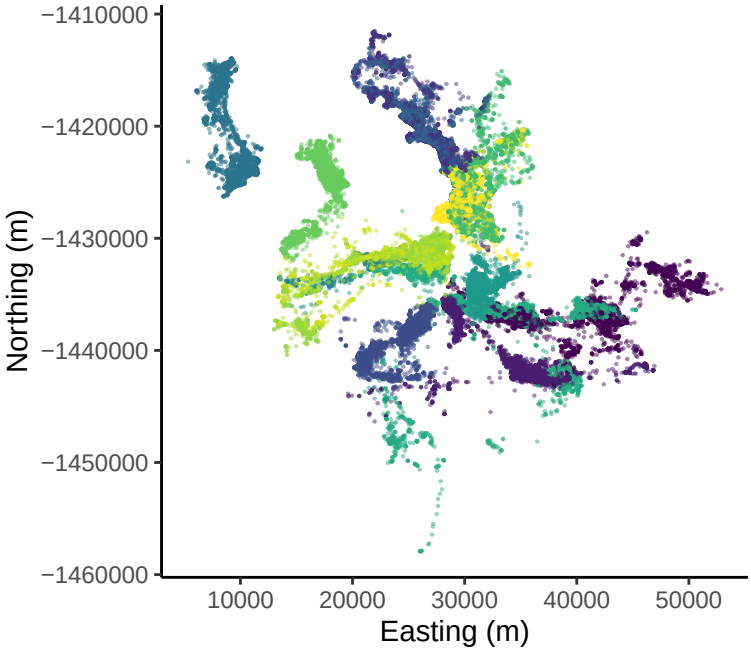
Create a step object

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

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

Plot the data spatially

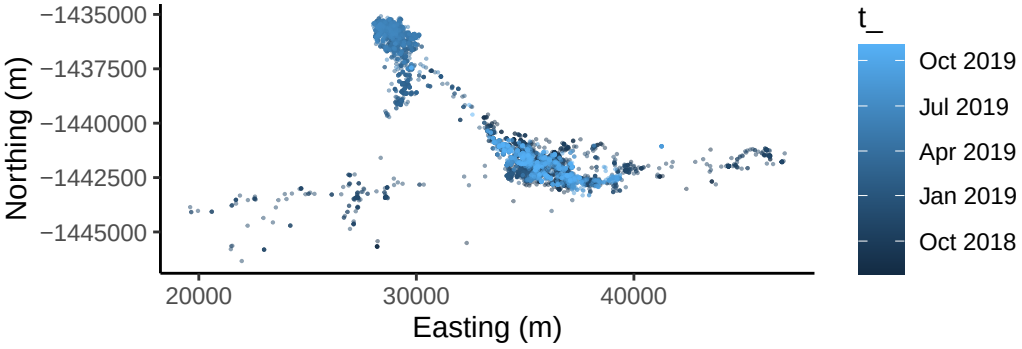
```
buffalo_all %>%
  ggplot(aes(x = x_, y = y_, colour = id)) +
  geom_point(alpha = 0.5, size = 0.1) +
  coord_fixed() +
  scale_x_continuous("Easting (m)") +
  scale_y_continuous("Northing (m)") +
  scale_colour_viridis_d() +
  theme_classic() +
  theme(legend.position = "none")
```




```
# ggsave("outputs/data_prep/buffalo_djelk_map.png",
#        width = 150, height = 150, units = "mm",  dpi = 600)
```

Pick out a single individual

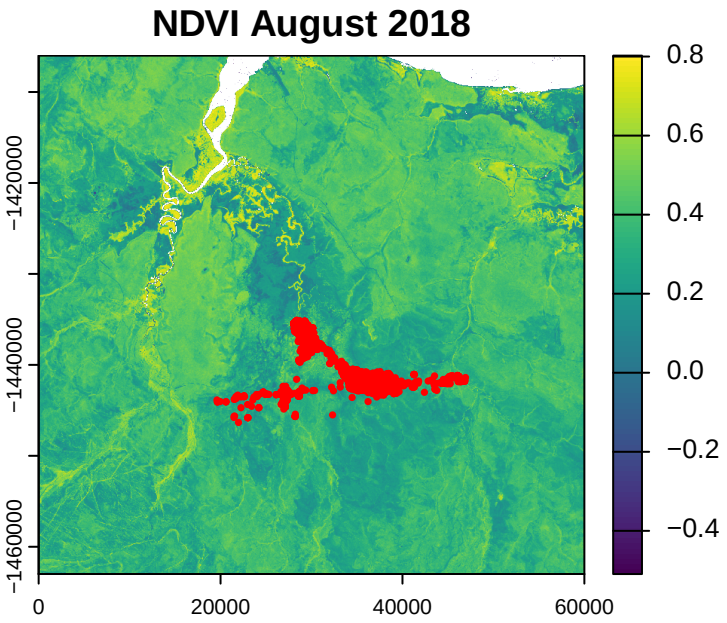
```
buffalo_id <- buffalo_all %>% filter(id == "2014")

buffalo_id %>%
  ggplot(aes(x = x_, y = y_, colour = t_)) +
  geom_point(alpha = 0.5, size = 0.1) +
  coord_fixed() +
  scale_x_continuous("Easting (m)") +
  scale_y_continuous("Northing (m)") +
  theme_classic()
```



Import spatial covariate

Although NDVI changes over time, and we have access to monthly layers, we will just select a single month here.

```
ndvi <- rast("mapping/ndvi_aug_2018.tif")
plot(ndvi, main = "NDVI August 2018")
points(buffalo_id$x_, buffalo_id$y_, col = "red", pch = 16, cex = 0.5)
```




```
ndvi
```

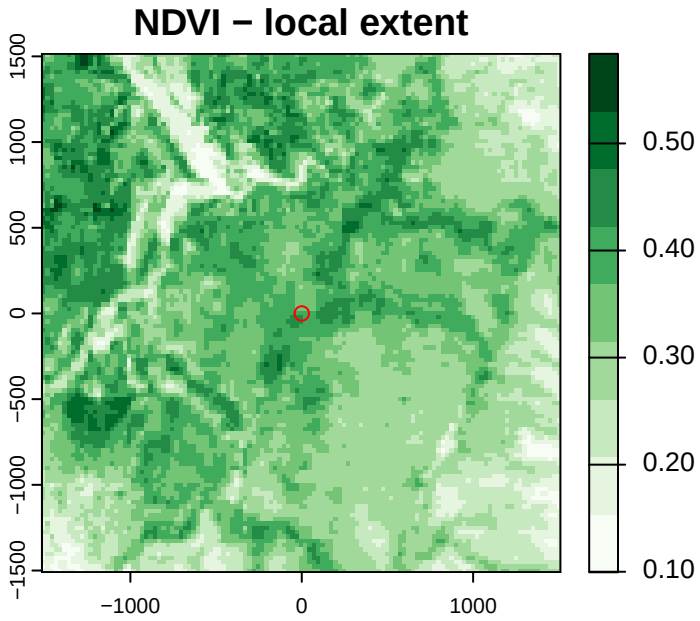
```
class       : SpatRaster
dimensions  : 2280, 2400, 1  (nrow, ncol, nlyr)
resolution  : 25, 25  (x, y)
extent      : 0, 60000, -1463000, -1406000  (xmin, xmax, ymin, ymax)
coord. ref. : GDA94 / Geoscience Australia Lambert (EPSG:3112)
source      : ndvi_aug_2018.tif
name        :       ndvi
min value   : -0.5441047
max value   :  0.8086554
```

Select local extent

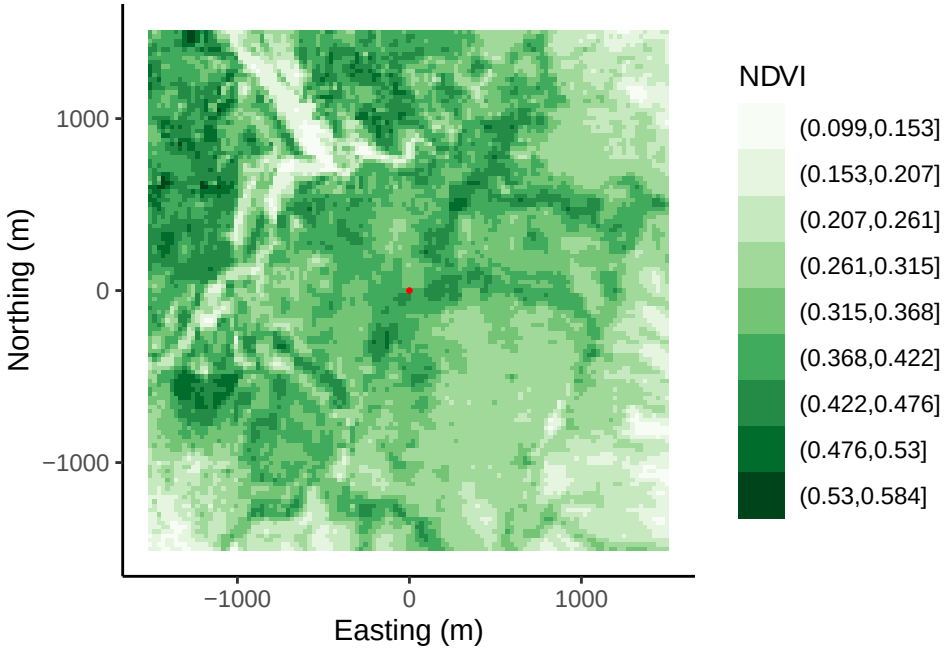
We just want to look at an area centred on a single location, which might be combined with our movement probability surface to generate a next-step probability surface.

We'll centre the extent on an interesting looking part of the landscape, and then buffer it by 1500m, which contains most of the step lengths, and therefore most of the movement kernel.

```
buffalo_single_point <- buffalo_id %>%
  filter(t_ == min(t_)) # select the first time point

buffer <- 1512.5 # buffer in metres

window_extent <- ext(buffalo_single_point$x_ - buffer,
                     buffalo_single_point$x_ + buffer,
                     buffalo_single_point$y_ - buffer,
                     buffalo_single_point$y_ + buffer)

ndvi_window <- crop(ndvi, window_extent)

# set the extent to 0 at the buffalo location
ndvi_window <- shift(ndvi_window,
                     dx = -buffalo_single_point$x_,
                     dy = -buffalo_single_point$y_)

# plot the NDVI layer with the buffalo location as the centre point
plot(ndvi_window, main = "NDVI - local extent",
     col = brewer.pal(9, "Greens"))
points(x = 0, y = 0, col = "red")
```



```r
# create a NDVI df to plot with ggplot
ndvi_df <- as.data.frame(ndvi_window, xy = TRUE)

ggplot() +
  geom_raster(data = ndvi_df, aes(x = x, y = y, fill = as.factor(cut(ndvi, breaks = 9)))) +
  scale_fill_brewer(palette = "Greens", direction = 1) +
  coord_fixed() +
  geom_point(aes(x = 0, y = 0),
             colour = "red", size = 0.5) +
  labs(x = "Easting (m)", y = "Northing (m)", fill = "NDVI") +
  theme_classic()
```



Sample random steps to fit models

```r
buffalo_id_steps <- buffalo_id %>%
  steps()

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

# movement parameters
gamma_dist$params$shape
```

```
[1] 0.3452605
```

```
gamma_dist$params$scale
```

```
[1] 1023.96
```

```
vonmises_dist$params$kappa
```

```
[1] 0.0483029
```

```
vonmises_dist$params$mu
```

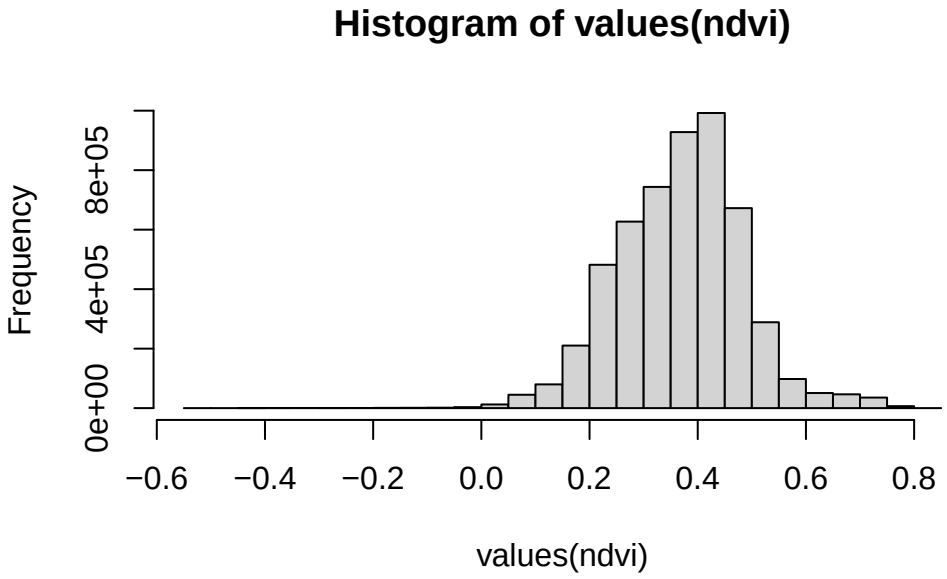
```
Circular Data:
Type = angles
Units = radians
Template = none
Modulo = asis
Zero = 0
Rotation = counter
[1] 0
```

```
# sample random steps
buffalo_ssf_data <- buffalo_id_steps %>%
  random_steps(n = 10,
               sl_dist = gamma_dist,
               ta_dist = vonmises_dist) %>%
  mutate(log_sl = log(sl_),
         cos_ta = cos(ta_)) %>%
  extract_covariates(ndvi)
```

Different SSF formulations

Firstly, we want to pull out the range of NDVI values to plot the curves with.

```
# histogram of NDVI values
hist(values(ndvi))
```




```
# 1% and 99% quantiles of NDVI values
ndvi_quantiles <- quantile(values(ndvi), probs = c(0.01, 0.99), na.rm = TRUE)
ndvi_quantiles
```

```
        1%        99%
0.09036818 0.68761911
```

```
# plot across the range in the local window
ndvi_values <- seq(ndvi_quantiles[1],
                   ndvi_quantiles[2],
                   length.out = 100)
```

Linear covariate

Model formula

$y \sim ndvi$

Fit model

```
# fit an issf model
ssf_linear <- amt::fit_issf(case_ ~
                              ndvi +
                              sl_ + log_sl + cos_ta +
```

```
                              strata(step_id_),
                            data = buffalo_ssf_data)

summary(ssf_linear)
```

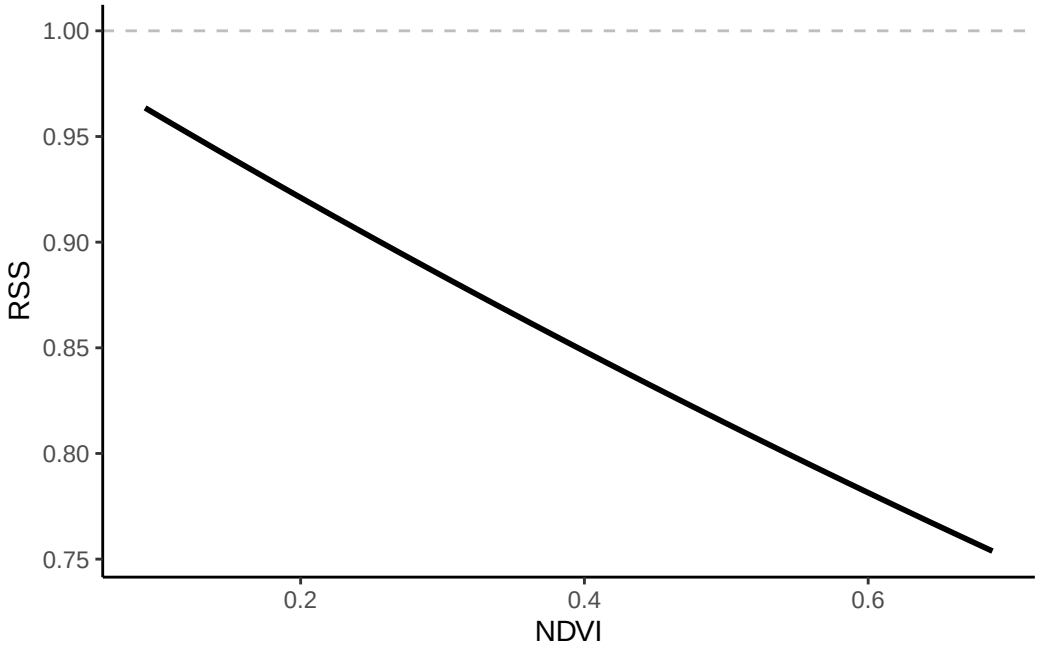
```
Call:
coxph(formula = Surv(rep(1, 72270L), case_) ~ ndvi + sl_ + log_sl +
    cos_ta + strata(step_id_), data = data, method = "exact")

  n= 72270, number of events= 6570

             coef  exp(coef)   se(coef)      z Pr(>|z|)
ndvi   -4.110e-01  6.630e-01  2.077e-01 -1.979   0.0479 *
sl_     3.229e-06  1.000e+00  2.514e-05  0.128   0.8978
log_sl -9.409e-04  9.991e-01  5.195e-03 -0.181   0.8563
cos_ta  2.760e-03  1.003e+00  1.824e-02  0.151   0.8797
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       exp(coef) exp(-coef) lower .95 upper .95
ndvi      0.6630     1.5084    0.4412    0.9961
sl_       1.0000     1.0000    1.0000    1.0001
log_sl    0.9991     1.0009    0.9889    1.0093
cos_ta    1.0028     0.9972    0.9676    1.0393

Concordance= 0.478  (se = 0.004 )
Likelihood ratio test= 3.98  on 4 df,   p=0.4
Wald test            = 3.97  on 4 df,   p=0.4
Score (logrank) test = 3.97  on 4 df,   p=0.4
```

Response curve

```
# extract the NDVI coefficient
ndvi_coef <- ssf_linear$model$coefficients[1]

# calculate the RSS for each NDVI value
ndvi_coef_df <- data.frame(ndvi_values,
                 "log_rss" = ndvi_values * ndvi_coef,
                 "rss" = exp(ndvi_values * ndvi_coef))

# plot the response curve
ggplot() +
```

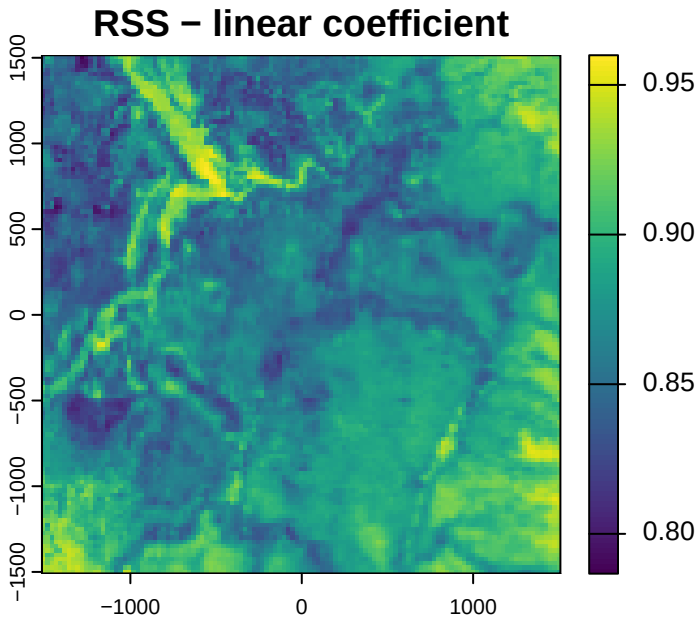
```
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey") +
  geom_line(data = ndvi_coef_df, aes(x = ndvi_values, y = rss),
            colour = "black", size = 1) +
  labs(x = "NDVI", y = "RSS") +
  theme_classic()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Plot habitat selection

```
# calculate the RSS for the local NDVI values
ndvi_linear <- exp(ndvi_window * ndvi_coef)

# plot the habitat selection (spatial RSS)
plot(ndvi_linear, main = "RSS - linear coefficient")
```



Quadratic covariate

Model formula

$y \sim ndvi + ndvi^2$

Fit model

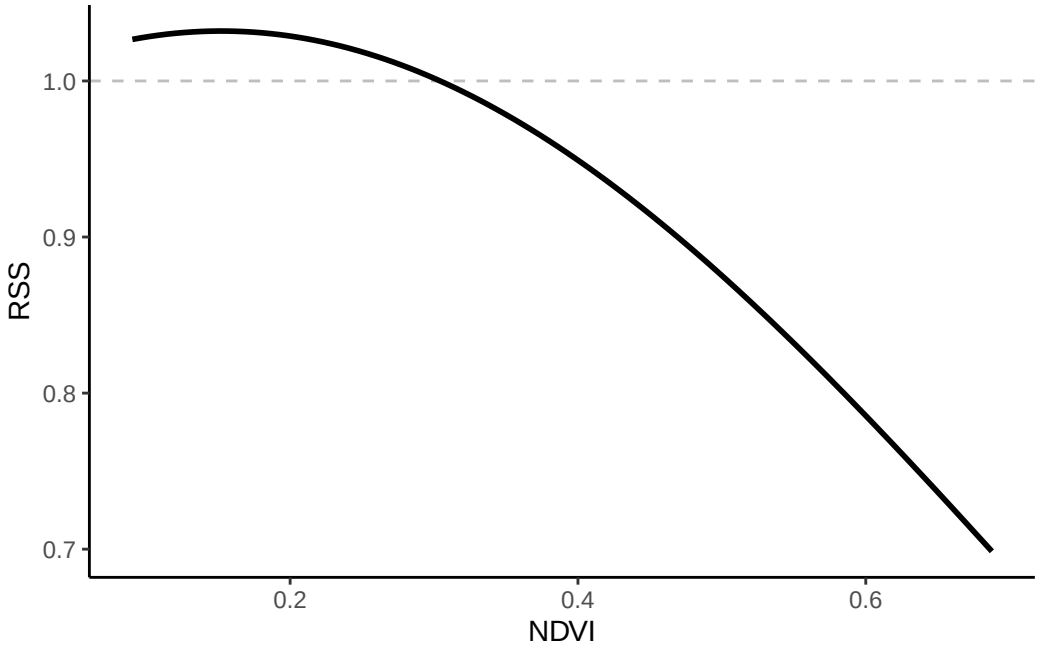
```
# fit an issf model
ssf_quadratic <- amt::fit_issf(case_ ~
                                 ndvi + I(ndvi^2) +
                                 sl_ + log_sl + cos_ta +
                               strata(step_id_),
                             data = buffalo_ssf_data)

summary(ssf_quadratic)
```

```
Call:
coxph(formula = Surv(rep(1, 72270L), case_) ~ ndvi + I(ndvi^2) +
    sl_ + log_sl + cos_ta + strata(step_id_), data = data, method = "exact")

  n= 72270, number of events= 6570

                coef  exp(coef)   se(coef)      z Pr(>|z|)
```

```
ndvi       4.139e-01  1.513e+00  9.290e-01  0.446    0.656
I(ndvi^2) -1.360e+00  2.565e-01  1.494e+00 -0.911    0.362
sl_        2.978e-06  1.000e+00  2.514e-05  0.118    0.906
log_sl    -9.433e-04  9.991e-01  5.196e-03 -0.182    0.856
cos_ta     2.792e-03  1.003e+00  1.824e-02  0.153    0.878

          exp(coef) exp(-coef) lower .95 upper .95
ndvi         1.5127     0.6611   0.24489     9.344
I(ndvi^2)    0.2565     3.8981   0.01372     4.795
sl_          1.0000     1.0000   0.99995     1.000
log_sl       0.9991     1.0009   0.98894     1.009
cos_ta       1.0028     0.9972   0.96759     1.039

Concordance= 0.48  (se = 0.004 )
Likelihood ratio test= 4.82  on 5 df,   p=0.4
Wald test            = 4.76  on 5 df,   p=0.4
Score (logrank) test = 4.76  on 5 df,   p=0.4
```

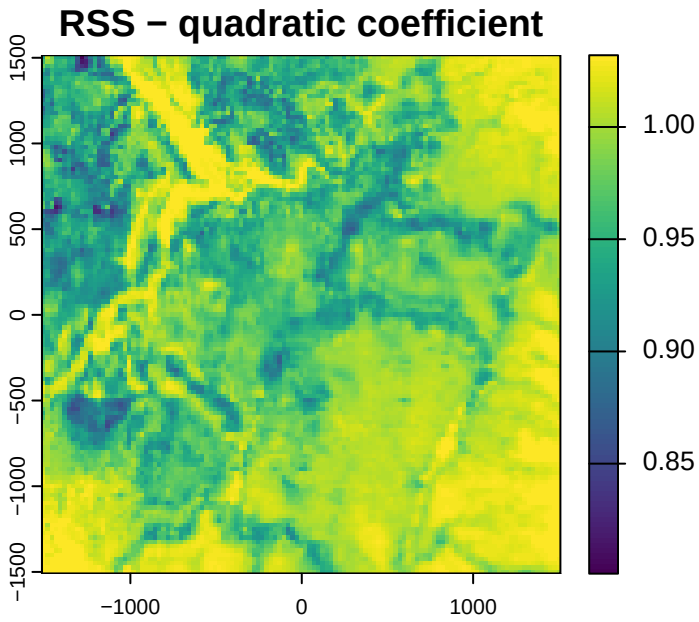
Response curve

```r
# extract the NDVI coefficient
ndvi_coef <- ssf_quadratic$model$coefficients[1]
ndvi_coef_sq <- ssf_quadratic$model$coefficients[2]

log_rss = (ndvi_values * ndvi_coef) + (ndvi_values^2 * ndvi_coef_sq)
rss = exp(log_rss)

# calculate the RSS for each NDVI value
ndvi_coef_df <- data.frame(ndvi_values, log_rss, rss)

# plot the response curve
ggplot() +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey") +
  geom_line(data = ndvi_coef_df, aes(x = ndvi_values, y = rss),
            colour = "black", size = 1) +
  labs(x = "NDVI", y = "RSS") +
  theme_classic()
```



Plot habitat selection

```
# calculate the RSS for the local NDVI values
ndvi_quadratic <- exp((ndvi_window * ndvi_coef) +
                        (ndvi_window^2 * ndvi_coef_sq))

# plot the habitat selection (spatial RSS)
plot(ndvi_quadratic, main = "RSS - quadratic coefficient")
```



Smooth term

Fit a GAM to the data, with a smooth term across the range of NDVI values (Klappstein et al. 2024).

Model formula

$y \sim s(ndvi)$

Fit model

```
# create a dummy variable create a dummy column for times (used in a Cox PH model, but not r
buffalo_ssf_data$times <- 1 # dummy time variable

# fit an issf model
ssf_gam <- mgcv::gam(cbind(times, step_id_) ~
                       # s(ndvi, bs = "tp", k = 10) +
                       s(ndvi) +
                       sl_ + log_sl + cos_ta,
                            data = buffalo_ssf_data,
                            family = cox.ph,
                            weight = case_)

summary(ssf_gam)
```

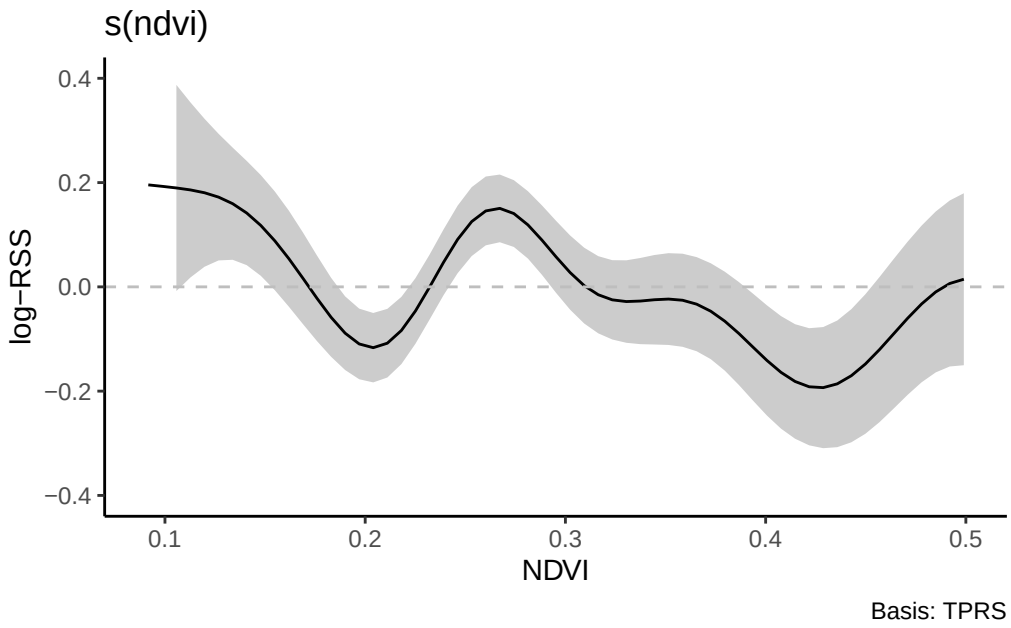
```
Family: Cox PH
Link function: identity

Formula:
cbind(times, step_id_) ~ s(ndvi) + sl_ + log_sl + cos_ta

Parametric coefficients:
        Estimate Std. Error z value Pr(>|z|)
sl_    3.116e-06  2.516e-05   0.124    0.901
log_sl 3.870e-04  5.210e-03   0.074    0.941
cos_ta 2.614e-03  1.824e-02   0.143    0.886

Approximate significance of smooth terms:
          edf Ref.df Chi.sq p-value
s(ndvi) 7.804  8.603  56.12  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Deviance explained = 0.0148%
-REML =  15759  Scale est. = 1         n = 72270
```

```r
# coef(ssf_gam)[4]

gam_smooths <- smooth_estimates(ssf_gam)
```

Response curve

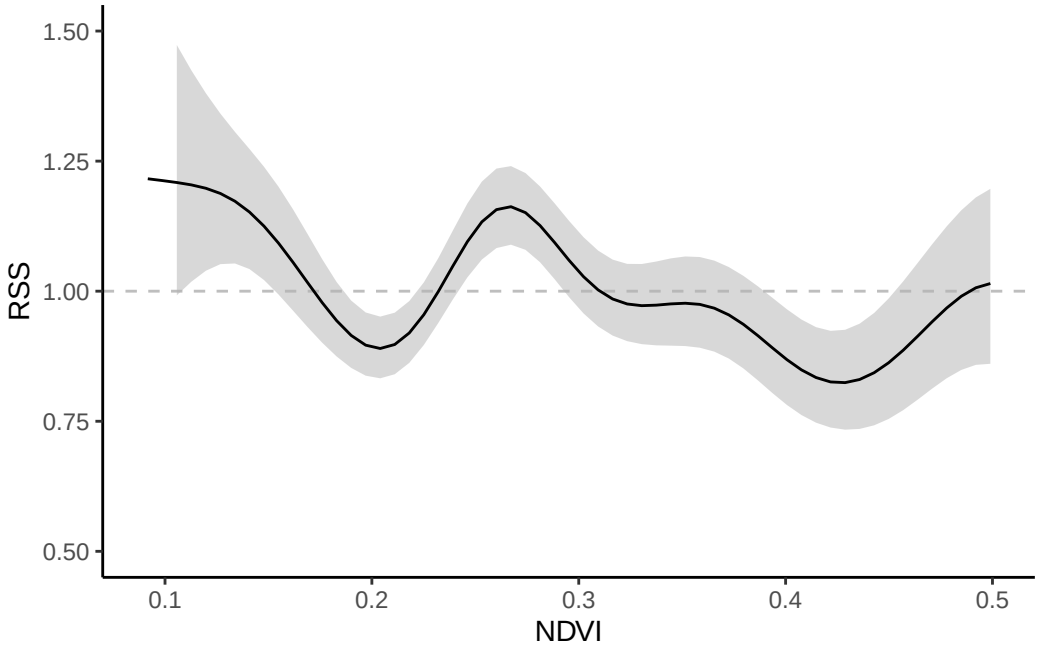
```r
gratia::draw(ssf_gam, rug = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed", colour = "grey") +
  labs(x = "NDVI", y = "log-RSS") +
  scale_x_continuous(limits = c(ndvi_quantiles[1], 0.5)) +
  scale_y_continuous(limits = c(-0.4, 0.4)) +
  theme_classic()
```

```
Warning: Removed 41 rows containing missing values or values outside the scale range
(`geom_line()`).
```



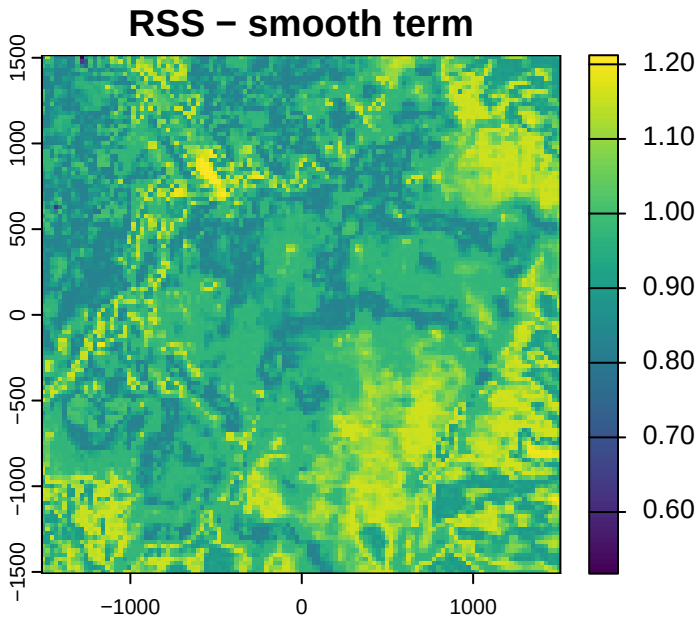

```
ggplot(data = gam_smooths) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey") +
  geom_ribbon(aes(x = ndvi, ymin = exp(.estimate - 1.96 * .se), ymax = exp(.estimate + 1.96
              fill = "grey70", alpha = 0.5) +
  geom_line(aes(x = ndvi, y = exp(.estimate)),
            colour = "black") +
  labs(x = "NDVI", y = "RSS") +
  scale_x_continuous(limits = c(ndvi_quantiles[1], 0.5)) +
  scale_y_continuous(limits = c(0.5, 1.5)) +
  theme_classic()
```

```
Warning: Removed 41 rows containing missing values or values outside the scale range
(`geom_line()`).
```



Plot habitat selection

```
predicted_log_rss <- predict(ssf_gam,
                         newdata = data.frame(ndvi = values(ndvi_window),
                                              sl_ = 0,
                                              log_sl = 0,
                                              cos_ta = 0),
                        type = "link")

ndvi_gam <- ndvi_window
values(ndvi_gam) <- exp(as.vector(predicted_log_rss))

# Plot the predictions
plot(ndvi_gam, main = "RSS - smooth term")
```



Klappstein, Natasha J, Théo Michelot, John Fieberg, Eric J Pedersen, and Joanna Mills Flemming. 2024. "Step Selection Functions with Non-linear and Random Effects." *Methods in Ecology and Evolution*, June. https://doi.org/10.1111/2041-210x.14367.