HMM-SSF example (part 2: fitting)

Natasha Klappstein

2023-01-16

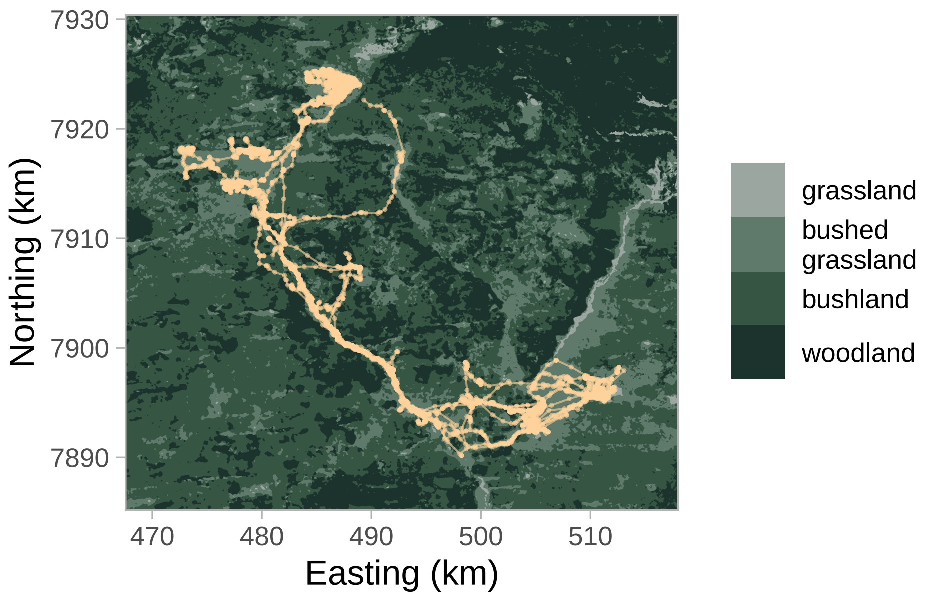
Contents

1 Introduction

This document shows a worked example of fitting HMM-SSF (to reproduce the analysis in the manuscript, "Flexible hidden Markov models for behaviour-dependent habitat selection", Klappstein et al. (https://doi.org/10.1101/2022.11.30.518554) or the file `HMM_SSF_bioRxiv.pdf`). Note, thaty this is a work-in-progress and most sections have incomplete details, and has not yet been thoroughly checked for errors, typos, etc. See the manuscript for further information and mathematical details of the model. Here, we start from a processed dataset with control locations and covariates; in this case, we are using the zebra example from the previous example and the manuscrupt. The dataset consists of half-hourly locations with step lengths, turning angles, vegetation/habitat type (grassland, bushed grassland, bushland, woodland), and time of day as covariates. We generated control locations with uniform turning angles and gamma-distributed step lengths (with parameters derived from the observed data).

```
data <- readRDS("data/zebra_processed.RData")
names(data)
```

```
##  [1] "obs"     "x"       "y"       "time"    "stratum" "ID"      "step"
##  [8] "angle"   "veg"     "tod"
```



2 Model formulation

We want to fit a two-state HMM-SSF, with both habitat selection (i.e., SSF) covariates, and covariates on the HMM transition probabilities. We explain the covariates in the next sections, and the model formulas can be specified in R as follows:

```
n_states <- 2
ssf_formula <- ~ step + log(step) + cos(angle) + veg
tpm_formula <- ~ cos(2*pi*tod/24) + sin(2*pi*tod/24)
```

2.1 SSF covariates

We model habitat selection based on habitat (i.e., vegetation type) and movement (i.e., steps and turns). We choose to model steps with a gamma distribution, and this can be accomplished by including step length and its log as covariates in the SSF. The von Mises is a circular distribution, naturally used for turning angles, and we include the cosine of the angle in the SSF. We include vegetation types as a categorical covariate with four levels: i) grassland (the reference category), ii) bushed grassland, iii) bushland, and iv) woodland. The habitat selection model for a step ending at y given that it started at x then takes the form,

$$f(y|x) \propto \exp\{\beta_1 L + \beta_2 \log(L) + \beta_3 \cos(\theta) + \beta_4 \delta^{b\_grass} + \beta_5 \delta^{bush} + \beta_6 \delta^{wood}\} \tag{1}$$

where L is the step length, θ is turning angle, and δ is an indicator variable for the categorical vegetation type variable (0 or 1).

2.2 Transition probability (HMM) covariates

We model transition probabilities based on time of day. We include a cyclic covariate, such that

$$\eta_{ij}^{(t)} = \begin{cases} \alpha_0^{(ij)} + \alpha_1^{(ij)} \cos\left(\frac{2\pi\tau_t}{24}\right) + \alpha_2^{(ij)} \sin\left(\frac{2\pi\tau_t}{24}\right) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

3 Model fitting

3.1 Initialisation

We use direct numerical optimisation of the likelihood via the forward algorithm for model fitting. This requires selecting initial parameter values for the optimiser. In general, for our functions, the initial SSF parameters for a model with K states and p covariates take the form,

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_1^{(1)} & \cdots & \beta_1^{(K)} \\ \beta_2^{(1)} & \cdots & \beta_2^{(K)} \\ \vdots & \ddots & \vdots \\ \beta_p^{(1)} & \cdots & \beta_p^{(K)} \end{pmatrix} \tag{3}$$

Similarly, the parameters for the transition probability matrix will have a row for each model term (intercept and all covariates), and include each transition. For example, in a 3-state model with p covariates, they take the form,

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_0^{(12)} & \alpha_0^{(13)} & \alpha_0^{(21)} & \alpha_0^{(23)} & \alpha_0^{(31)} & \alpha_0^{(32)} \\ \alpha_1^{(12)} & \alpha_1^{(13)} & \alpha_1^{(21)} & \alpha_1^{(23)} & \alpha_1^{(31)} & \alpha_1^{(32)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_p^{(12)} & \alpha_p^{(13)} & \alpha_p^{(21)} & \alpha_p^{(23)} & \alpha_p^{(31)} & \alpha_p^{(32)} \end{pmatrix} \tag{4}$$

The initial parameters must be carefully chosen to ensure we find the global minima (rather than local) of the likelihood function. This can be challenging, and it is common to try a range of starting values, ultimately selecting the fitted model with the lowest negative log likelihood. We don't show this in detail here, but in practice, we tried multiple starting values: movement initial values based on a fitted HMM, paired with several different habitat selection initial values. The `illustration` folder of this repository has the full code to do so: `initial_par.R` defines the sets of initial values, and `zebra_fit.R` fits the model with all sets and selects the best model. Here, we just load the initial parameters and only fit the model with the best set:

```r
initial_par <- readRDS("code/illustration/initial_par.RData")
initial_par <- initial_par[[3]]
initial_par
```

```
## $betas
##              [,1]       [,2]
## [1,] -12.1197731 -3.1176507
## [2,]  -1.1209529 -0.5031641
## [3,]   0.1008005  1.7585377
## [4,]  -2.0000000  2.0000000
## [5,]  -2.0000000  2.0000000
## [6,]  -2.0000000  2.0000000
##
## $alphas
##      [,1] [,2]
## [1,]   -2   -2
## [2,]    0    0
## [3,]    0    0
```

Note, the initial parameters are stored in a list with matrices for the β (i.e., SSF) and α (i.e., HMM) parameters in the form specified above (but note the change in dimension of $\boldsymbol{\alpha}$).

3.2 Calling the optimisation function

We use a custom fitting function `fitHMMSSF`, which calls the `R` function `optim`. `fitHMMSSF` takes the following arguments:

- `ssf_formula`: habitat selection model formula in the form `~step_covariates + cos(angle) + habitat_covariates`
- `tpm_formula`: transition probability model (set to `~1` if no covariates)
- `data`: data processed with control locations and all necessary covariates
- `par0`: initial parameters, in a list form with matrices called `betas` and `alphas`
- `n_states`: how many states in the HMM
- `dist`: control step length distribution (gamma or uniform)

```
fit <- fitHMMSSF(ssf_formula = ssf_formula,
                 tpm_formula = tpm_formula,
                 data = data,
                 par0 = initial_par,
                 n_states = n_states,
                 dist = "gamma",
                 optim_opts = list(trace = 1,
                                   maxit = 1e4))
```

In this case, the model properly converged in 7161 optimisation iterations. The fitted model parameters are returned (with 95% confidence intervals) in a list with the following elements:

- `betas`: estimated habitat (i.e., SSF) coeffcients with 95% CIs (`lower' andupper`), returned with named covariates and states
- `alphas`: estimated transition probability covariates with 95% CIs (`lower' andupper`), returned with named covariates and transitions
- `convergence`: convergence code returned from `optim` (0 indicates successful convergence, but see `optim` documentation for all codes)
- `nllk`: the negative log-likelihood
- `hessian`: the estimated Hessian matrix returned from `optim`

The fitted mode outputs take the following forms:

```
fit$betas
```

```
##    state                cov     estimate        lower        upper
## 1      1               step -13.57513095 -14.82515916 -12.32510275
## 2      1          log(step)  -1.18076535  -1.21457832  -1.14695238
## 3      1         cos(angle)  -0.01694447  -0.07340508   0.03951614
## 4      1 vegbushed grassland   0.21039575   0.02974241   0.39104908
## 5      1        vegbushland  -0.19119272  -0.45743085   0.07504541
## 6      1        vegwoodland  -0.56131124  -1.07615003  -0.04647246
## 7      2               step  -3.11298132  -3.36153845  -2.86442418
## 8      2          log(step)  -0.65347195  -0.77619718  -0.53074672
## 9      2         cos(angle)   1.45893447   1.32769409   1.59017485
## 10     2 vegbushed grassland  -0.99785993  -1.16488311  -0.83083676
## 11     2        vegbushland  -2.18767791  -2.41726452  -1.95809130
## 12     2        vegwoodland  -1.96101593  -2.26089024  -1.66114161
```

```
fit$alphas
```

```
##   transition                cov    estimate       lower      upper
## 1        1-2        (Intercept) -1.83464051 -1.98099590 -1.6882851
## 2        1-2 cos(2 * pi * tod/24)  0.03326221 -0.13964765  0.2061721
## 3        1-2 sin(2 * pi * tod/24)  0.81074958  0.63444599  0.9870532
## 4        2-1          (Intercept) -1.25631748 -1.39582637 -1.1168086
## 5        2-1 cos(2 * pi * tod/24)  0.15436559 -0.02192779  0.3306590
## 6        2-1 sin(2 * pi * tod/24)  0.14069751 -0.03833573  0.3197308
```

```
fit$convergence
```

```
## [1] 0
```

```
fit$nllk
```

```
## [1] -12611.3
```

4 Inference

The estimated model parameters tell us about habitat selection and movement patterns, as well as behavioural state dynamics. We start by explaining state decoding (i.e., state classification) and then interpret the SSF and HMM covariates.

4.1 State decoding

State decoding is useful to understand the most likely states along the time series. There are two types of state decoding: global (estimated with the Viterbi algorithm) and local (estimated with the forward-backward algorithm). Global decoding focuses on the most likely state sequence, whereas local decoding derives the local state probability for each observation. The outputs are generally quite similar, but not identical, as they solve different optimisation problems.

To obtain decoded states, we use two custom functions: i) `viterbi_decoding` and ii) `local_decoding`. Both functions require the SSF and transition probability model formulas, the data, the fitted model, and the number of states. Global decoding returns the most likely sequence of states, and local decoding returns the local probabilities of each state.

```
# global decoding via the Viterbi algorithm
viterbi <- viterbi_decoding(ssf_formula = ssf_formula,
                            tpm_formula = tpm_formula,
                            data = data,
                            fit = fit,
                            n_states = n_states)

# view output (i.e., state sequence)
head(viterbi, 100)
```

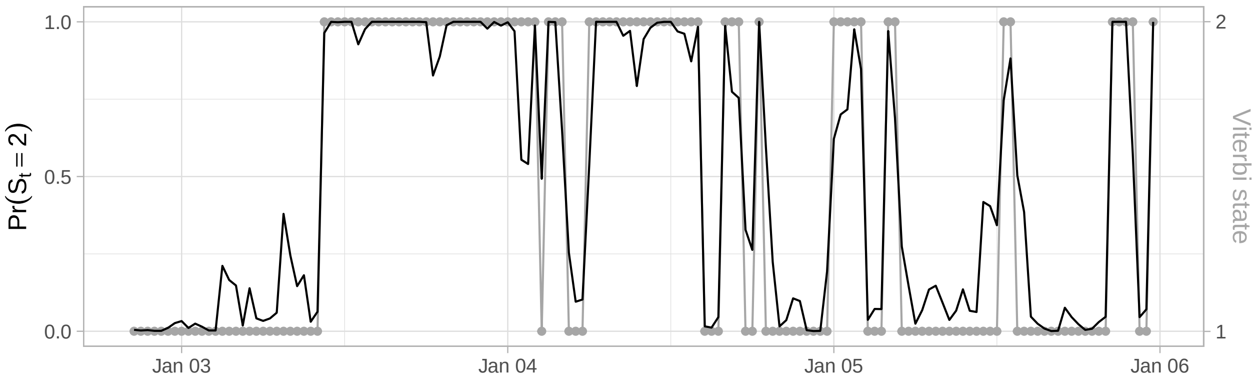
```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1
##  [75] 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
```

```
# local decoding via the forward-backward algorithm
sp <- local_decoding(ssf_formula = ssf_formula,
                     tpm_formula = tpm_formula,
                     data = data,
                     fit = fit,
                     n_states = n_states)

# view output (i.e., local probabilities of each state)
head(sp)
```

```
##         state1      state2
## [1,] 0.7218096 0.278190412
## [2,] 0.7112208 0.288779241
## [3,] 0.6951980 0.304802037
## [4,] 0.9055759 0.094424137
## [5,] 0.9943285 0.005671478
## [6,] 0.9725931 0.027406910
```

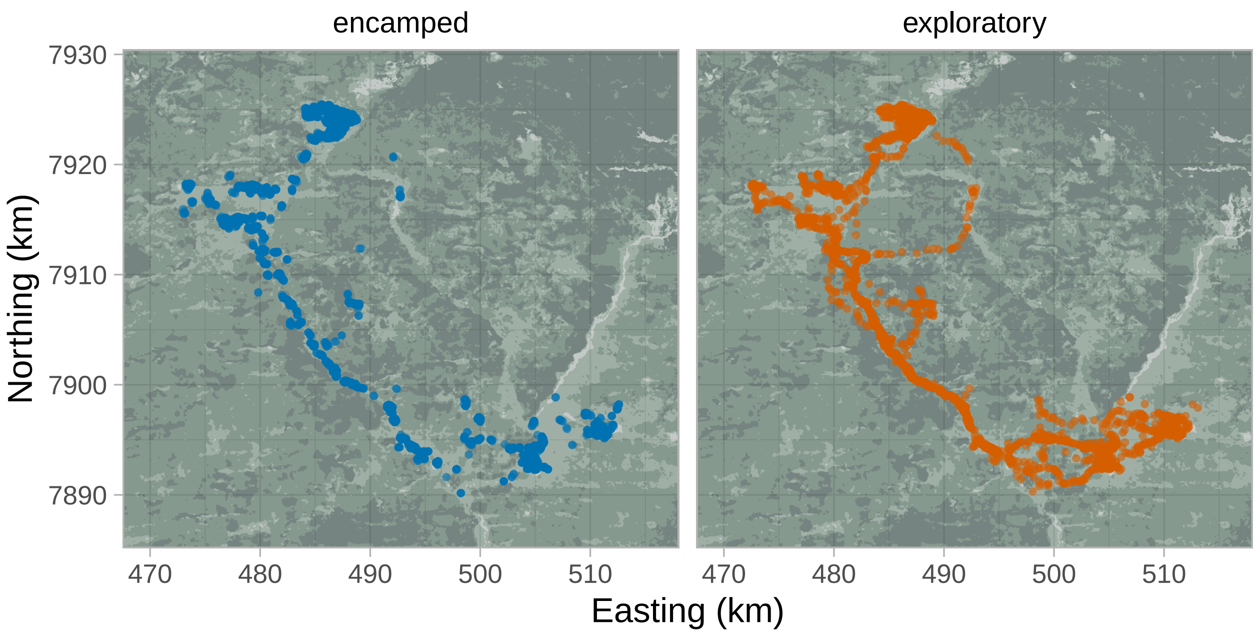
We can see how these two outputs compare (for a subset of the time series for visualation purposes). Note, here we are plotting both on the same graph, but they have distinct axes. The Viterbi sequence takes values of 1 or 2, while the local state probabilities are between (0,1).



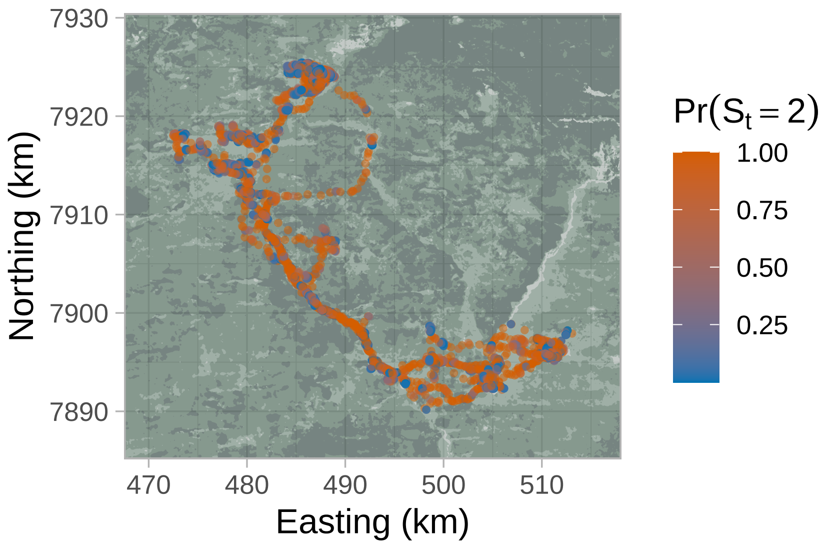
The outputs generally agree, and there are very few locations where the outputs predict a diffrent state. In general, the local state probabilities give more information about the uncertainty in state classification, whereas the Viterbi sequence only gives the estimated state.

4.1.1 Spatial visualisation of decoded states

We can visual the decoded states spatially, by subsetting the observations by the states of the Viterbi sequence and plotting them separately:



Similarly, we can obtain a plot of the local state probabilities by plotting the observations, coloured by the probability (could be in either state 1 or 2; here we plot for state 2):



4.2 SSF parameters

The estimate SSF parameters tell us about habitat selection and movement patterns.

4.2.1 Movement estimates

From the fitted model, we can estimate the distributions of steps and turns, but the raw parameter estimates cannot easily be interpreted.

```
fit$betas[which(fit$betas$cov == "step" |
                  fit$betas$cov == "log(step)" |
                  fit$betas$cov == "cos(angle)"),]
```

```
##   state        cov     estimate        lower        upper
## 1     1       step -13.57513095 -14.82515916 -12.32510275
## 2     1  log(step)  -1.18076535  -1.21457832  -1.14695238
## 3     1 cos(angle)  -0.01694447  -0.07340508   0.03951614
## 7     2       step  -3.11298132  -3.36153845  -2.86442418
## 8     2  log(step)  -0.65347195  -0.77619718  -0.53074672
## 9     2 cos(angle)   1.45893447   1.32769409   1.59017485
```

Remember, we fitted the model assuming that steps followed a gamma distribution, and we can derive the distribution parameters from the estimated β_1 and β_2 for each state. We can derive the shape a and scale b paramters from the following relationships: $\beta_1 = -1/b$ and $\beta_2 = a - 2$. The gamma distribution can also be parameterised in terms of its mean μ and standard deviation σ, which is more interpretable. We do so via the following,

$$\mu = -\frac{\beta_2 + 2}{\beta_1} \quad \text{and} \quad \sigma = -\frac{\sqrt{\beta_2 + 2}}{\beta_1}\ . \tag{5}$$

These are both implemented with the following functions:

```
get_shape_scale(fit, n_states = 2)
```

```
##   state     shape      scale
## 1     1 0.8192346 0.07366411
## 2     2 1.3465281 0.32123546
```

```
beta_to_mean(fit, n_states = 2)
```

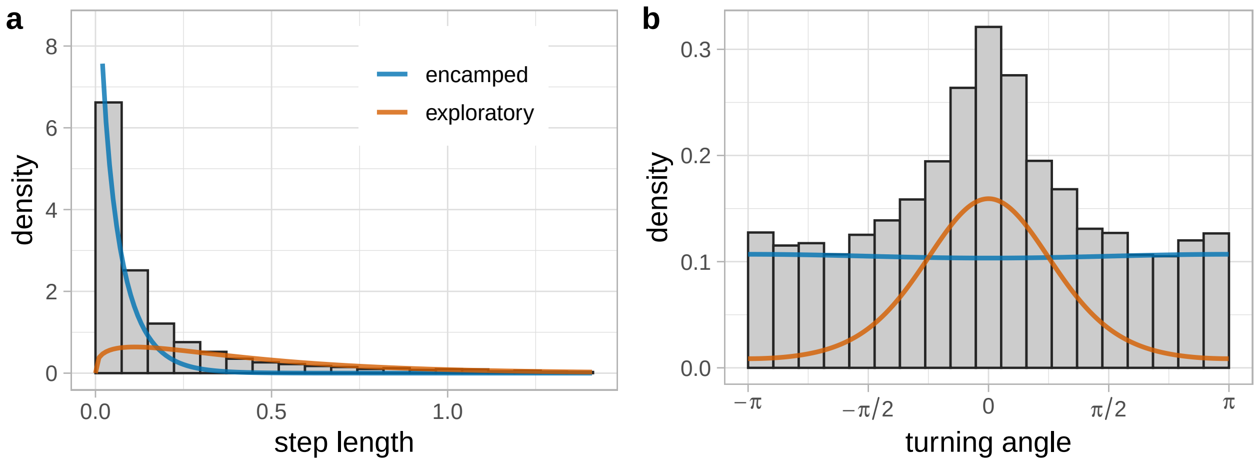
```
##   state       mean         sd
## 1     1 0.06034819 0.06667456
## 2     2 0.43255256 0.37276162
```

From this, we can see that the zebra moved faster in state 2. Similarly, we modelled turning angles θ with a von Mises distribution by including the $\cos(\theta)$ as a covariate. The estimated β_3 for each state is the angular concentration parameter of the von Mises distribution. The mean can either be 0 (if the parameter is positive) or π (if the parameter is negative), which indicate directional persistence or reversion (respectively).

```
fit$betas[which(fit$betas$cov == "cos(angle)"),]
```

```
##   state        cov    estimate       lower      upper
## 3     1 cos(angle) -0.01694447 -0.07340508 0.03951614
## 9     2 cos(angle)  1.45893447  1.32769409 1.59017485
```

Notice here that $\beta_3^{(1)}$ is negative (although CIs overlap 0) and $\beta_3^{(2)}$ is positive. This indicates that the zebra has a very weak tendency to revert direction in the first, slower state and a stronger tendency to persist in direction when travelling faster in state 2. From here on, we refer to state 1 as "encamped" behaviour and state 2 as "exploratory" behaviour. We can plot the estimate step length and turning distributions for each state. Note, these plots are made by estimating the distributions, weighted by the number of observations estimated to be in each state (via the Viterbi algorithm). The step length axes are both truncated for visualisation purposes.



4.2.2 Habitat selection estimates

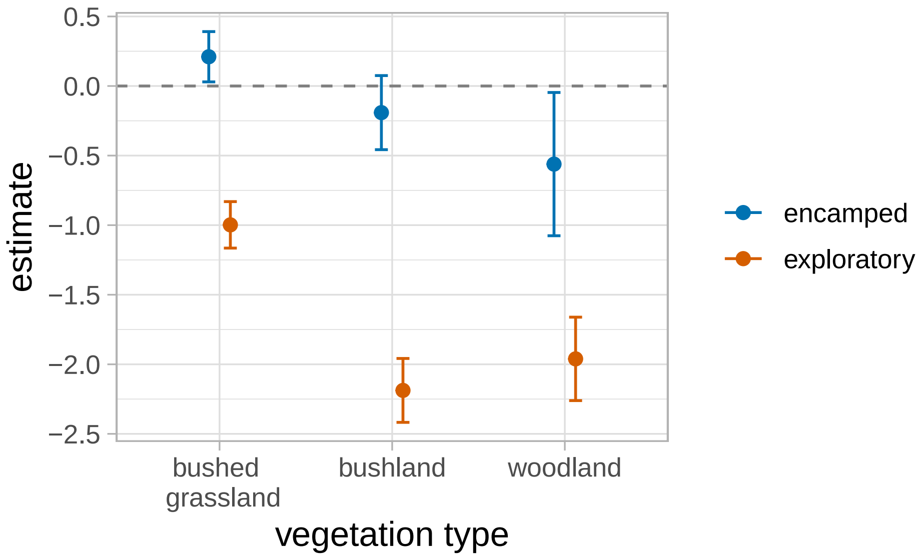
The remaining SSF parameters represent habitat selection for different vegetation types.

```
habitat_coef <- fit$betas[which(fit$betas$cov == "vegbushed grassland" |
                  fit$betas$cov == "vegbushland" |
                  fit$betas$cov == "vegwoodland"),]

habitat_coef
```

```
##    state                 cov   estimate       lower       upper
## 4      1 vegbushed grassland  0.2103957  0.02974241  0.39104908
## 5      1         vegbushland -0.1911927 -0.45743085  0.07504541
## 6      1         vegwoodland -0.5613112 -1.07615003 -0.04647246
## 10     2 vegbushed grassland -0.9978599 -1.16488311 -0.83083676
## 11     2         vegbushland -2.1876779 -2.41726452 -1.95809130
## 12     2         vegwoodland -1.9610159 -2.26089024 -1.66114161
```

A negative selection coefficient indicates avoidance of that vegetation type (compared to the reference category), and positive coefficients indicate selection for that vegetation type. We can plot these with uncertainty, where the dashed line at 0 indicates no selection. Here, we can see that the zebra selects against all vegetation types (compared to grassland) in both states, with the excption of selection for bushed grassland in the encamped state.



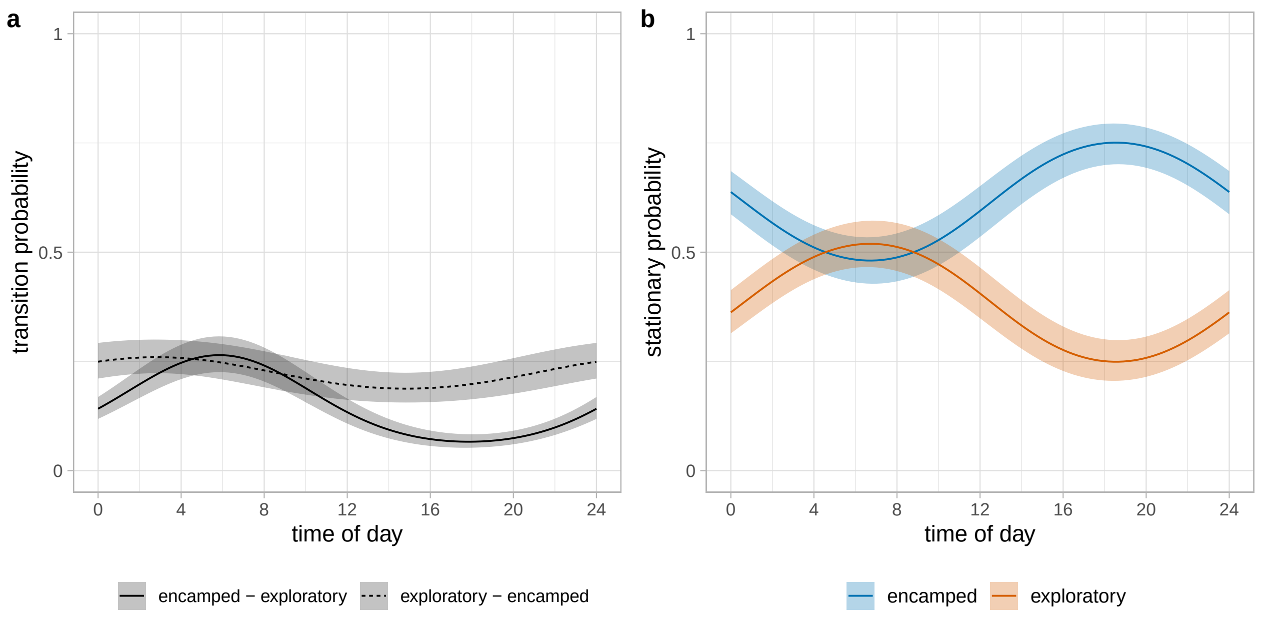
Another common way to interpret SSF parameter estimates is in terms of their relative selection strength (RSS). This can be derived as the exponential of the estimate, and this represents how much more likely the zebra is to take a step in each habitat type compared to the reference category grassland. Since nearly all coefficients are negative, we may alternatively be interested to derive the RSS of grassland, which the 1/RSS:

```
habitat_coef$grass_RSS <- 1/exp(habitat_coef$estimate)
habitat_coef
```

```
##    state                 cov   estimate       lower       upper grass_RSS
## 4      1 vegbushed grassland  0.2103957  0.02974241  0.39104908 0.8102635
## 5      1         vegbushland -0.1911927 -0.45743085  0.07504541 1.2106928
## 6      1         vegwoodland -0.5613112 -1.07615003 -0.04647246 1.7529696
## 10     2 vegbushed grassland -0.9978599 -1.16488311 -0.83083676 2.7124707
## 11     2         vegbushland -2.1876779 -2.41726452 -1.95809130 8.9144888
## 12     2         vegwoodland -1.9610159 -2.26089024 -1.66114161 7.1065431
```

4.3 Transition probability parameters

We also want to make inference about the temporal dynamics of the transition probabilities. This section will be finished soon.


a
b
1
0.5
0
transition probability
stationary probability
0
4
8
12
16
20
24
time of day
encamped − exploratory
exploratory − encamped
encamped
exploratory