State-switching step selection functions in hmmSSF

Natasha Klappstein & Théo Michelot

2023-06-08

Contents

We describe the analysis workflow for the R package hmmSSF, which implements state-switching step selection functions as described by Klappstein et al. [2023] (and originally proposed by Nicosia et al. [2017]).

1 Data preparation

We consider an example (simulated) track with 2500 locations from one animal, and one spatial covariate. The tracking data is shown on top of a heatmap of the covariate in Figure @ref(fig:plot-data).

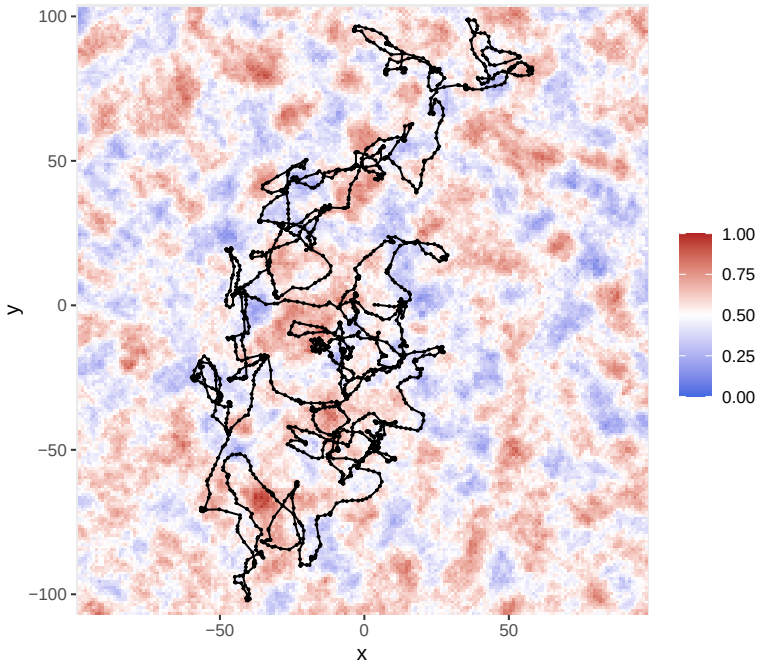


Figure 1: Example data.

1.1 Format of tracking data

The package expects data to be provided as a data frame with the following columns:

- `ID`: identifier for track/animal
- `x`: Easting
- `y`: Northing
- `time`: time of observation

The locations need to be projected from longitude-latitude to Easting-Northing prior to

analysis, as the package uses Euclidean (planar) geometry to derive metrics such as step lengths and turning angles.

```
head(track)
```

```
  ID         x          y                time
1  1  3.517854  0.1967671 2020-01-01 00:00:00
2  1  4.781500  0.1262261 2020-01-01 01:00:00
3  1  8.635494 -0.2937678 2020-01-01 02:00:00
4  1 10.328140 -0.5135970 2020-01-01 03:00:00
5  1 11.852726 -0.1078853 2020-01-01 04:00:00
6  1 12.090271 -0.9006765 2020-01-01 05:00:00
```

1.2 Generating control steps

The most common method to fit a step selection function is to generate a set of "control" steps for each observed step, to weigh the suitability of the chosen move against alternatives. Forester et al. [2009] pointed out that it is advantageous to generate the control steps from distributions of movement variables that resemble the observed movement pattern. Klappstein et al. [2023] describe this as a form of importance sampling.

In hmmSSF, the control steps are generated using the function `get_controls()`, with multiple possible distributions (estimated from the observed data). The sampling distributions are specified by the argument `distr`, with the following options:

- if `distr = "uniform"`, then the control steps are generated by sampling points uniformly on a disc around each observed location, with radius set to the maximum observed step length.
- if `distr` is set to `"gamma"` or `"exp"`, then this is used as a distribution of distances (estimated from the distribution of observed step lengths) to generate control steps, and the turning angles are uniform.
- if `distr` is specified as a vector of two character strings, then the first is used as distribution of distances (`"gamma"` or `"exp"`), and the second as a distribution of turning angles (`"vm"` or `"wrpcauchy"`) to generate control steps.

The number of control steps also needs to be specified, with larger numbers leading to better approximations but higher computational cost. In general, the choice of the distribution and number of control locations is difficult, as it can affect the results, but there is no general guidance. We encourage users to test different distributions and numbers of control locations, to assess the sensitivity of their inferences.

For this example, we decide to generate 50 control locations per observed step, using a gamma distribution of distances. The output has the same columns as the original data frame, and a few new ones: `stratum` identifies the stratum to which a location belongs, and `obs` is a binary variable indicating whether a location is observed or not.

```
data <- random_locs(obs = track, n_controls = 50, distr = "gamma")

head(data)
```

```
  ID stratum obs        x          y       step       angle                time
1  1       3   1 8.635494 -0.2937678 3.87681125 -0.05278238 2020-01-01 02:00:00
2  1       3   0 5.675062 -0.6194955 1.16385268 -0.63969012 2020-01-01 02:00:00
3  1       3   0 5.143816  0.8498971 0.80930400  1.16238281 2020-01-01 02:00:00
4  1       3   0 1.847962  2.1092420 3.54090345  2.60293491 2020-01-01 02:00:00
5  1       3   0 4.835666  0.1277922 0.05418846  0.08467172 2020-01-01 02:00:00
6  1       3   0 3.789019  0.1708322 0.99348242 -3.13074097 2020-01-01 02:00:00
```

In the importance sampling approach, the control steps need to be weighted during model fitting based on how likely they were under the distribution used to generate them. This technical detail is hidden in hmmSSF, and the weights are automatically computed and stored as an attribute of the data frame (for later use in `fitHMMSSF()`). For this reason, the control steps need to have been generated by `get_controls()`, rather than some other method (e.g., the amt package).

1.3 Extracting spatial covariates

The spatial covariate needs to be evaluated at all observed and control locations. The raster is stored as a `SpatRaster`, and we use the package terra for this purpose.

```
data$cov1 <- extract(cov1, data[,c("x", "y")])$layer
```

2 Model specification and model fitting

The main modelling decisions are the choice of the number of states in the HMM, and the choice of the SSF formula within each state.

There is no general method to select the best number of states, and we recommend leaning towards fewer states (2 or 3) in most applications to avoid numerical problems. (See also Pohle et al. [2017] for a discussion of this issue.) Here, we use two states, which we hope can separate slow and fast movement phases.

The formula should include movement variables as well as covariates (Avgar et al. [2016]). For example, including step length (and possibly its log) captures the preference for some range of movement speeds, and including the cosine of turning angle captures directional persistence. See Avgar et al. [2016] and Klappstein et al. [2023] for more details. We use a formula with step length, log step length, cosine of turning angle, and the spatial covariate `cov1`.

```
# number of HMM states
n_states <- 2

# SSF formula
ssf_formula <- ~ step + log(step) + cos(angle) + cov1
```

In hmmSSF, the model is fitted through numerical likelihood optimisation, and starting values need to be chosen for the model parameters. The choice of these values does not affect the model specification, but poorly chosen values can lead to numerical problems (e.g., failure of optimiser to converge). The general idea is that, if starting values are close to the "true" parameters, it will be easier for the optimiser to converge. In practice, the true parameters are not known, and so our best bet is to choose plausible values given the data. Here, we use the fact that the selection parameters for `step` and `log(step)` are linked to parameters of a gamma distribution, and the parameter for `cos(angle)` is the concentration of a von Mises distribution (Klappstein et al. [2023]).

The parameters should be passed as a matrix, with one row for each covariate in the formula (here, 4), and one column for each state (here, 2).

```
# initial values for SSF parameters
ssf_par0 <- matrix(c(-3, -1,
                     -1, 2,
                     0.2, 5,
                     3, -1),
                   ncol = 2, byrow = TRUE)
```

Finally, we can pass the model formulation, the data, and the starting parameter values to the function `fitHMMSSF()` for model fitting.

```
# fit model
mod <- fitHMMSSF(ssf_formula = ssf_formula,
                 n_states = n_states,
                 data = data,
```

```
                  ssf_par0 = ssf_par0)
```

The fitted model object can then be printed to find the estimated parameter values.

```
mod
```

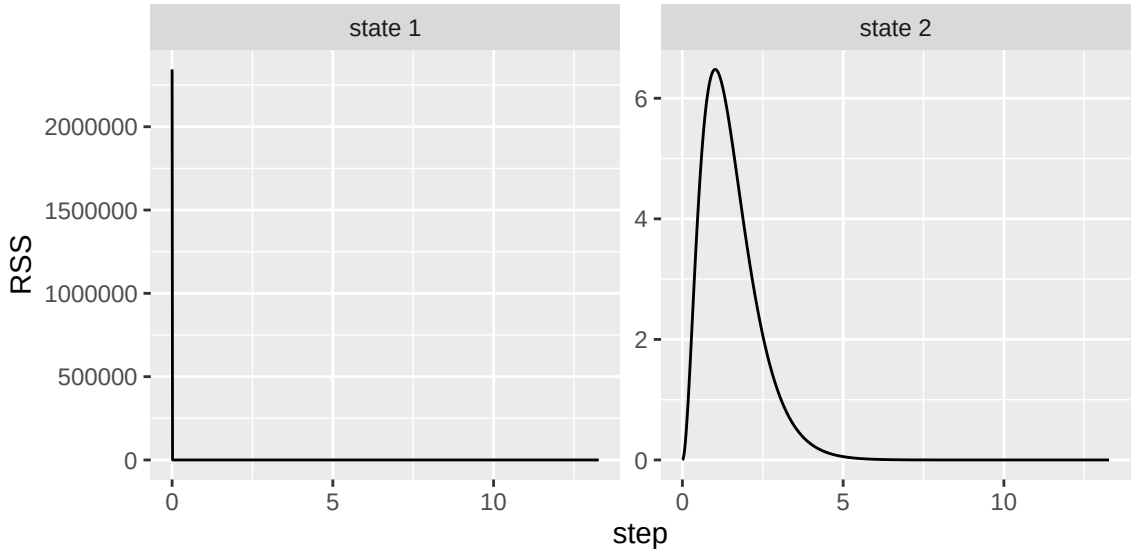
```
Negative log-likelihood: -1340.382
Convergence code: 0

SSF model:
~step + log(step) + cos(angle) + cov1
                  mle    low    upp
step.S1        -3.722 -4.069 -3.374
log(step).S1   -0.986 -1.054 -0.918
cos(angle).S1   0.204  0.128  0.281
cov1.S1         3.171  1.217  5.124
step.S2        -2.019 -2.221 -1.816
log(step).S2    2.059  1.660  2.458
cos(angle).S2   4.653  4.242  5.064
cov1.S2        -1.643 -2.949 -0.338

TPM model:
~1
      S1    S2
S1 0.903 0.097
S2 0.131 0.869
```

3 Interpretation of results

The function `plot_ssf()` can be used to plot the step selection function against any covariate included in the model. **Important:** the y axis shows relative likelihoods of selection, and y values should only be interpreted by comparing two covariate values. More precisely, for some covariate x (chosen as `var` argument) with estimated selection parameter β, the plot shows $k \times \exp(\beta x)$ against a grid of x values. The multiplicative constant k is arbitrary, which is why the scale of the y axis should only be interpreted in a relative way.

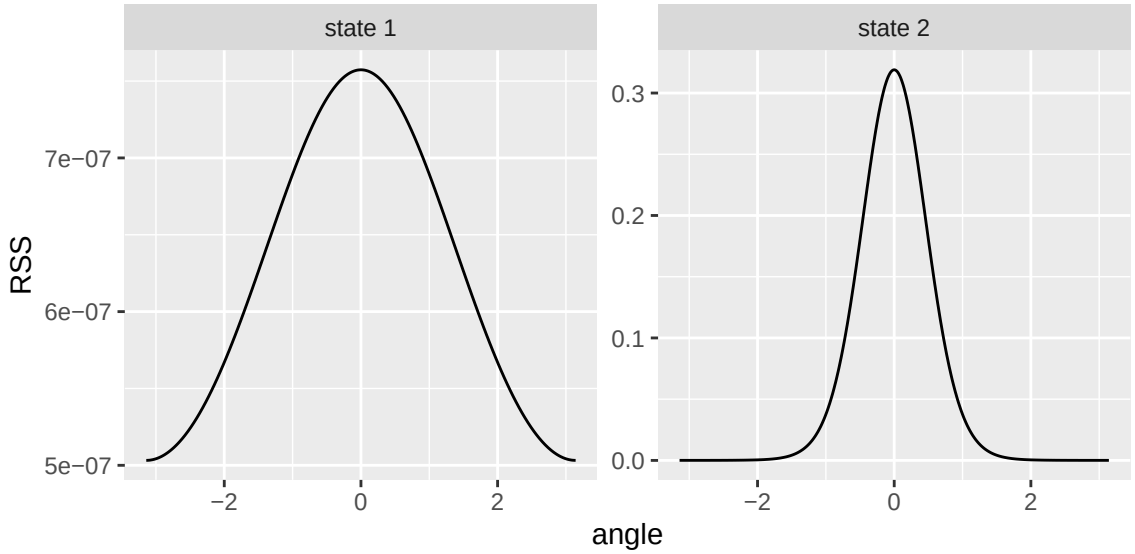
We first plot selection for step length L, $\exp(\beta_1 L + \beta_2 \log(L))$, in each state. The two distributions of step length are very different: state 1 captures shorter steps, and state 2 longer steps.

```
plot_ssf(mod = mod, var = "step")
```



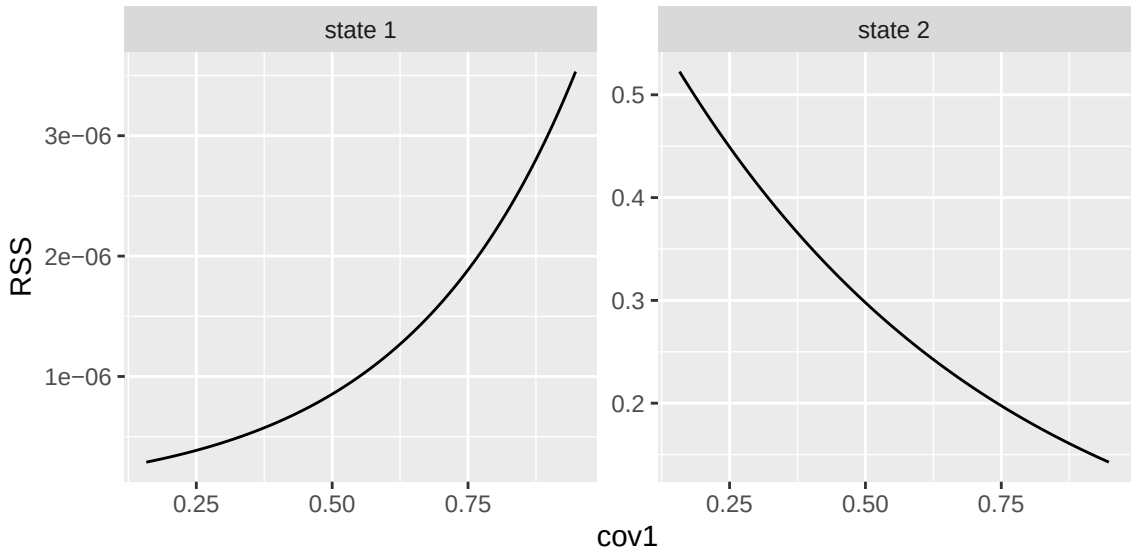
Similarly, we can plot selection for turning angle α, i.e, $\exp(\beta_3 \cos(\alpha))$. Movement in both state tends to be directed (i.e., with a peak at zero), and the directionality is stronger in state 2. This is consistent with the expectation that fast movement also tends to be more directed.

```
plot_ssf(mod = mod, var = "angle")
```



Finally, we plot the selection for the spatial covariate `cov1`, i.e., $\exp(\beta_4 x)$. The results suggest that the animal selected for the covariate in state 1 (positive relationship), but tended to avoid it in state 2 (negative relationship).

```
plot_ssf(mod = mod, var = "cov1")
```



Another useful output is the most likely state sequence, which can be computed with the function `viterbi_decoding()`. We can use it to visualise the movement track coloured by states, which confirms that states 1 and 2 correspond to slow and fast movement, respectively.

```
# get most likely state sequence
states <- viterbi_decoding(mod = mod)

# data frame for plotting (remove first two obs - not used in model)
df <- data.frame(x = track$x[-(1:2)],
                 y = track$y[-(1:2)],
                 state = factor(states))

ggplot(df, aes(x, y, col = state, group = NA)) +
  geom_path() +
  geom_point(size = 0.2) +
  coord_equal()
```

4 Covariates on transition probabilities

The effects of covariates on the transition probabilities of the hidden process can be included with the argument `tpm_formula`. Klappstein et al. [2023] describes this model, and discusses when a variable should be included in this part of the model (rather than in the SSF directly, say).

References

Tal Avgar, Jonathan R Potts, Mark A Lewis, and Mark S Boyce. Integrated step selection analysis: bridging the gap between resource selection and animal movement. *Methods in Ecology and Evolution*, 7(5):619–630, 2016.

James D Forester, Hae Kyung Im, and Paul J Rathouz. Accounting for animal movement in estimation of resource selection functions: sampling and data analysis. *Ecology*, 90(12): 3554–3565, 2009.

Natasha Jean Klappstein, Len Thomas, and Theo Michelot. Flexible hidden markov models for behaviour-dependent habitat selection. *Movement Ecology*, 11(1):1–13, 2023.

Aurélien Nicosia, Thierry Duchesne, Louis-Paul Rivest, and Daniel Fortin. A multi-state conditional logistic regression model for the analysis of animal movement. *Annals of Applied Statistics*, 2017.

Jennifer Pohle, Roland Langrock, Floris M Van Beest, and Niels Martin Schmidt. Selecting the number of states in hidden Markov models: pragmatic solutions illustrated using animal movement. *Journal of Agricultural, Biological and Environmental Statistics*, 22:270–293, 2017.