# Supplementary material for "Hidden Markov models: pitfalls and opportunities in ecology"

Richard Glennie[1], Timo Adam[1], Vianey Leos-Barajas[2], Théo Michelot[1], Theoni Photopoulou[1], and Brett T. McClintock[3]

[1]Centre for Research into Ecological and Environmental Modelling, University of St Andrews, St Andrews, KY16 9LZ, UK

[2]Department of Statistical Sciences, University of Toronto, Toronto, ON, Canada

[3]Marine Mammal Laboratory, NOAA-NMFS Alaska Fisheries Science Center, Seattle, USA

4th November 2021

## S1 Continuous-time HMM simulation study

### S1.1 Petrel analysis with multistate Brownian motion

This simulation study is based on realistic parameter values, obtained from the analysis of a real data set of Antarctic petrels (*Thalassoica antarctica*) from the Movebank data repository (Descamps et al., 2016a,b). The code for this analysis is included as supplementary material, and we describe it briefly here. We first converted the locations from longitude-latitude to Easting-Northing (measured in kilometres), and then kept only the first trajectory of the data set to decrease the computational cost of the analysis.

We modelled the location $X_t$ of the animal with a 2-state Brownian motion model, and we denote $(x_1, \dots, x_n)$ the observations at times $t_1 < \cdots < t_n$. The likelihood of each movement step from $x_i$ to $x_{i+1}$ is assumed to depend on the state at time $t_{i+1}$, i.e.,

$$X_{t_{i+1}} | \{X_{t_i} = x_i, S_{t_{i+1}} = j\} \sim N(x_i, \sigma_j \Delta_i), \tag{S1}$$

where $\Delta_i = t_{i+1} - t_i$. The $\sigma_j$ are diffusion parameters, related to the speed of movement of the animal. Here, we note that the multistate Brownian motion does not satisfy the snapshot property, because the distribution of $X_{t_{i+1}}$ really depends on the state process over the whole interval $[t_i, t_{i+1})$, rather than only at the time of observation $t_{i+1}$. Fitting this model as a continuous-time HMM is therefore an approximation, and the associated error is evaluated in the simulations.

This continuous-time HMM is defined by a latent 2-state continuous-time Markov chain (the state process), and by an observation model given in Equation S1. There were therefore four parameters to estimate in this analysis: the two transition rates $q_{12}$ and $q_{21}$ of the latent state process, and the two diffusion parameters $\sigma_1$ and $\sigma_2$ of the observation process. We fitted this model as a continuous-time HMM using the forward algorithm (Zucchini et al., 2017), and found the parameter estimates given in the main text.

## S1.2 Simulation procedure

We simulated time series from a model which violates the snapshot property (state-switching Brownian motion), and then tried to recover the model parameters using a continuous-time HMM (i.e., under the assumption that the snapshot property is satisfied). For different time intervals $\Delta \in \{0.25, 0.5, 1, \ldots, 16\}$ (in hours), we repeated the following steps 200 times:

(1) Generate 2000 irregular observation times uniformly from $[0, 2000\Delta]$, i.e., such that the mean time interval between observations is $\Delta$, and sort them to obtain a time grid $t_1 < t_2 < \cdots < t_{2000}$.

(2) Simulate a continuous-time 2-state Markov chain from $t_1$ to $t_{2000}$ to get the times of behavioural switches, with transition rates $q_{12}$ and $q_{21}$.

(3) Simulate a 2-state Brownian motion between $t_1$ and $t_{2000}$ with diffusion parameters $(\sigma_1, \sigma_2)$. To do this, we augmented the observation times with the switching times, such that the state is fixed over each time interval, and then simulated Brownian motion over each interval with the appropriate diffusion parameter.

(4) Exclude the switching times from the simulated data and only retain the data simulated at the observation times $t_1, \ldots, t_{2000}$ (similarly to a real scenario where the switching times are not known).

(5) Fit a 2-state HMM to the remaining observations to recover $\sigma_1$, $\sigma_2$, $\lambda_{12}$ and $\lambda_{21}$.

(6) Estimate the hidden states at the times of observations using the Viterbi algorithm.

# S2 Hierarchical HMM simulation study

## S2.1 Further details on the simulation procedure

In this section, we provide further details on the procedure of the simulation experiment outlined in Section 3.2.

Over 200 replications, we conducted the following steps:

(1) We generated 1000 realisations from a 2-state coarse-scale Markov chain with transition probability matrix

$$\mathbf{\Gamma} = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}.$$

Conditional on the simulated coarse-scale states, we draw 1000 coarse-scale observations of step length from a gamma distribution with state-dependent means $\mu_1 = 5, \mu_2 = 20$ and variances

$\sigma_1 = 4, \sigma_2 = 8$ and 1000 observations of turning angle from a von Mises distribution with mean 0 and state-dependent concentrations $\kappa_1 = 0.5, \kappa_2 = 5$. Coarse-scale state 1 thus captured small, undirected steps, while coarse-scale state 2 captured large, directed steps.

(2) For each simulated coarse-scale state, we generated 100 realisations from a 2-state fine-scale Markov chain with transition probability matrix

$$\mathbf{\Gamma}^{(k)*} = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix},$$

$k = 1, 2$. The simulated fine-scale states were then progressively shifted by $0, 5, 10, 15$, and 20 observations, such that the assumption of aligned state processes is violated. Conditional on the shifted fine-scale states, we draw 100 fine-scale observations of acceleration from a normal distribution with state-dependent means $\mu_1^{(1)*} = 1, \mu_2^{(1)*} = 3, \mu_1^{(2)*} = 2, \mu_2^{(2)*} = 4$ and variances $\sigma_1^{(1)*} = 0.5, \sigma_2^{(1)*} = 0.25, \sigma_1^{(2)*} = 0.25, \sigma_2^{(2)*} = 0.5$.

To assess the consequences of such a violation of the dependence structure, we computed the percentage bias as $(\hat{\theta}_i/\theta - 1) \cdot 100$, where $\hat{\theta}_i$ denotes the estimate for $\theta$ obtained in the $i^{\text{th}}$ replication.

## S2.2 Full results from the simulation experiment

In this section, we provide full results from the simulation experiment outlined in Section 3.2.

Fig. S1 displays the percentage bias obtained across all 200 replications. The parameters associated with the coarse-scale process, which are displayed in Fig. S1 (a), (d), and (e), are not affected by the shifting of the fine-scale process. However, the parameters associated with the fine-scale process become biased as the shifting progresses: the transition probabilities associated with the two fine-scale HMMs (Fig. S1 (b) and (c)) are, on average, biased by about 4 % (regardless of whether the

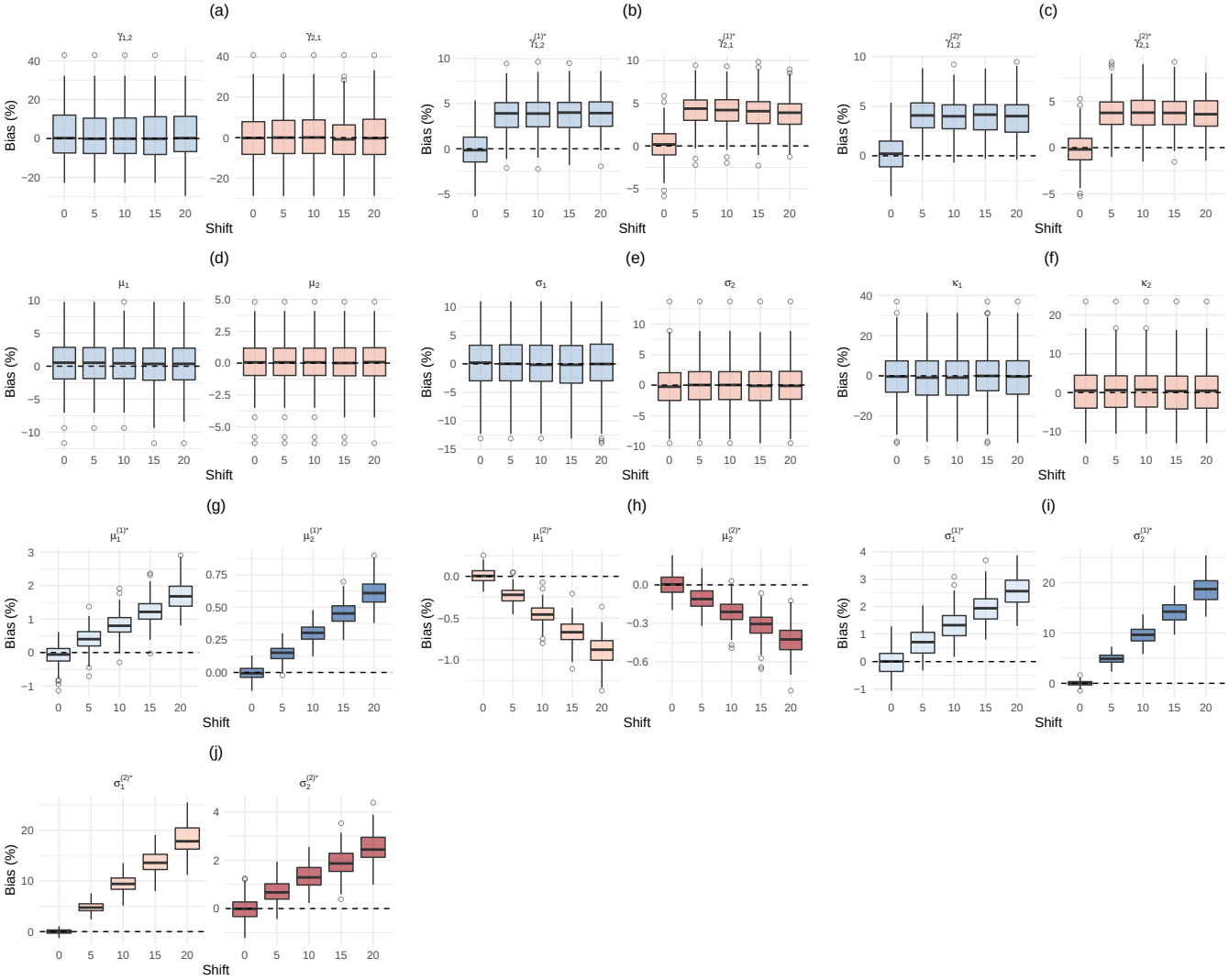



**Fig. S1.** Full results from the simulation experiment. Displayed is the percentage bias obtained across all 200 replications. The transition probabilities of the coarse-scale state process are denoted by $\gamma_{i,j}$ (a); the transition probabilities associated with fine-scale HMM $k$ are denoted by $\gamma_{i,j}^{(k)*}$ ((b) and (c)). The means of the step lengths under state $i$ are denoted by $\mu_i$ (d); the corresponding variances are denoted by $\sigma_i$ (e). The concentrations of the turning angles under state $i$ are denoted by $\kappa_i$ (f). The means of the accelerations under state $i$ associated with fine-scale HMM $k$ are denoted by $\mu_i^{(k)*}$ ((g) and (h)); the corresponding variances are denoted by $\sigma_i^{(k)*}$ ((i) and (j)).

fine-scale process was shifted by 5 or 20 observations), which indicates that the persistence within the fine-scale states is underestimated (or, in other words, the estimates suggest more switching between the fine-scale states than there is in the true data-generating process). The bias in the

means (Fig. S1 (f) and (g)) and variances (Fig. S1 (h) and (i)) of the accelerations increases as the shifting progresses, where the largest bias is observed for the variances.This severe bias is due to the fact that each of the two fine-scale HMMs must accommodate observations within each hour that truly belong to the alternate fine-scale HMM: a restriction imposed by having an hourly coarse-scale process.

# S3 Random Effects

## S3.1 Simulation 1

We present a simulation to demonstrate potential pitfalls with the inclusion of random effects in the observation process of an HMM when analyzing time series collected across multiple individuals. Let $K = 20$ indicate the number of individuals, $T = 100$ be the length of each time series, with $y_{t,k}$ denoting the $t^{th}$ observation from individual $k$ for $t \in \{1, \ldots, T\}$ and $k \in \{1, \ldots, K\}$. For an $N$-state HMM, assume the state-dependent distributions follow a normal distribution, i.e. $f_n(y_{t,k}) \sim Normal(\mu_n, \sigma_n)$, with $E(y_{t,k}|S_{t,k} = n) = \mu_n$ and $Var(y_{t,k}|S_{t,k} = n) = \sigma_n^2$. One manner to allow for variation across individuals in the observation process is to allow for individual-specific state-dependent means so that $E(y_{t,k}|S_{t,k} = n) = \mu_{k,n}$. We can further make the assumption of a population-level state-dependent mean, $\mu_{k,n} \sim N(\mu_n, \tau_n)$.

For this simulation, let

$$\mu_{k,1} \sim N(0, 0.1) \quad \mu_{k,2} \sim N(2, 0.1) \quad \mu_{k,3} \sim N(5, 0.1)$$

$$\sigma_1 = 0.3 \quad \sigma_2 = 1 \quad \sigma = 1.5$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.7 & 0.1 \\ 0.05 & 0.3 & 0.65 \end{bmatrix}$$

$$\boldsymbol{\delta} = [1/3, 1/3, 1/3]$$

The $K \times N$ state-dependent distributions are shown in Figure S2.

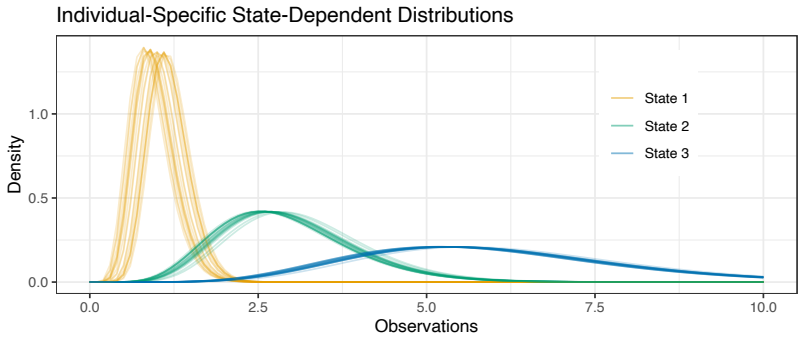



**Fig. S2.** Simulated state-dependent distributions across 20 time series.

We fit the model to the simulated data in a Bayesian framework using the software `Stan`. Priors were given as, $\boldsymbol{\Gamma}_{i,\cdot} \sim Dirichlet(\mathbf{1})$, $\boldsymbol{\delta} \sim Dirichlet(\mathbf{1})$, $\boldsymbol{\sigma} \sim N^{+}(0.5, 1)$, $\boldsymbol{\mu} \sim N(2, 3)$, $\boldsymbol{\tau} \sim N^{+}(0.1, 0.3)$, with a further ordering of the population and individual-specific means, i.e. $\mu_1 < \mu_2 < \mu_3$ and $\mu_{k,1} < \mu_{k,2} < \mu_{k,3}$, for $k \in \{1, \ldots, K\}$.

As demonstrated in Figure S3, fitting the correctly specified model to the generated data does not necessarily imply that the individual-specific state-dependent densities will be captured perfectly. The results for Time Series 4 show that state 1 is captured adequately, while both the means of

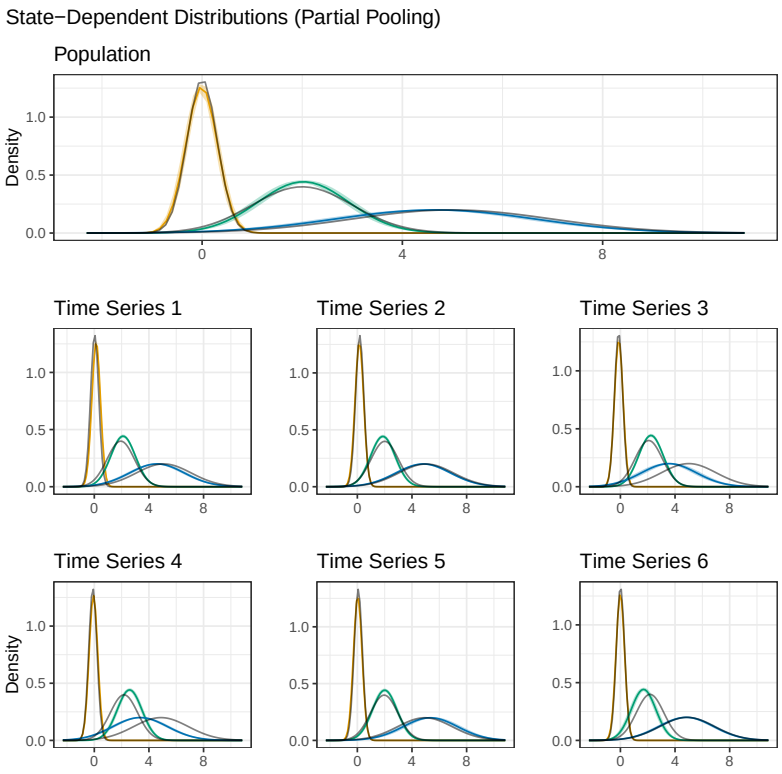



**Fig. S3.** Estimates of state-dependent distributions for the population and first six time series along with 95% pointwise credible intervals with the true values in grey.

state 2 and 3 are, respectively, under and overestimated. Similarly for other time series we can see that the state-dependent distributions are not always captured perfectly.

The estimated transition probability matrix and 95% credible intervals also demonstrate a lack of fit in terms of the state-switching dynamics of state 2 and 3.

$$\hat{\mathbf{\Gamma}} = \begin{bmatrix} \mathbf{0.79}(0.78, 0.80) & \mathbf{0.10}(0.10, 0.11) & \mathbf{0.10}(0.09, 0.11) \\ \mathbf{0.21}(0.20, 0.23) & \mathbf{0.66}(0.65, 0.68) & \mathbf{0.12}(0.11, 0.14) \\ \mathbf{0.04}(0.04, 0.05) & \mathbf{0.26}(0.24, 0.28) & \mathbf{0.70}(0.68, 0.72) \end{bmatrix}$$

## S3.2 Garter Snakes

Additional information for the garter snake analysis. For the 3-state HMM, priors were given as, $\boldsymbol{\Gamma}_{i,\cdot} \sim Dirichlet(\mathbf{1})$, $\boldsymbol{\delta} \sim Dirichlet(\mathbf{1})$, $\boldsymbol{\sigma} \sim T_3^+(0,1)$, $\boldsymbol{\mu} \sim N(3,1)$, $\boldsymbol{\tau} \sim T_3^+(0,1)$, with a further ordering of the population and individual-specific means, i.e. $\mu_1 < \mu_2 < \mu_3$ and $\mu_{k,1} < \mu_{k,2} < \mu_{k,3}$, for $k \in \{1, \ldots, K\}$.

# S4 Continuous state spaces

```
# Packages needed

library(Matrix)

library(ggplot2)

library(numDeriv)

library(pathintegrateR)

# remotes::install_github("r-glennie/pathintegrateR")
```

In this appendix, we provide an introduction to implementing spatial hidden Markov models (HMMs) with diffusion and state-switching animal movement models (Pedersen et al., 2008; Thygesen et al., 2009; Pedersen et al., 2011a). We then introduce how advection processes can be incorporated and the shortcomings with the current, most popular approaches.

We intend for this tutorial to be a practical, short introduction to those unfamiliar with the implementation of spatial HMMs, leading to their further use in practice and as an opening for statistical researchers to progress development in these methods.

For this tutorial, we will be working in continuous time.

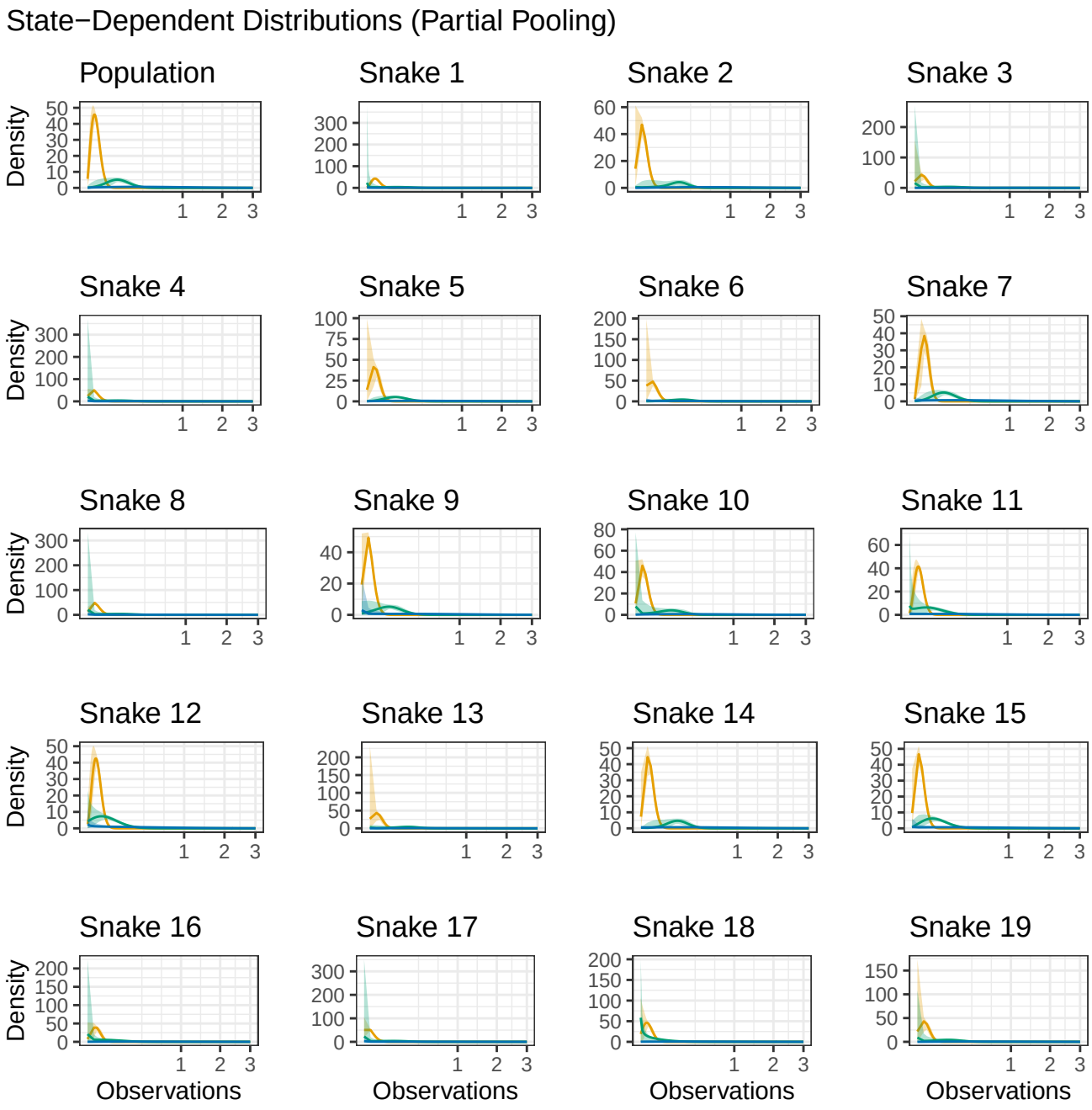


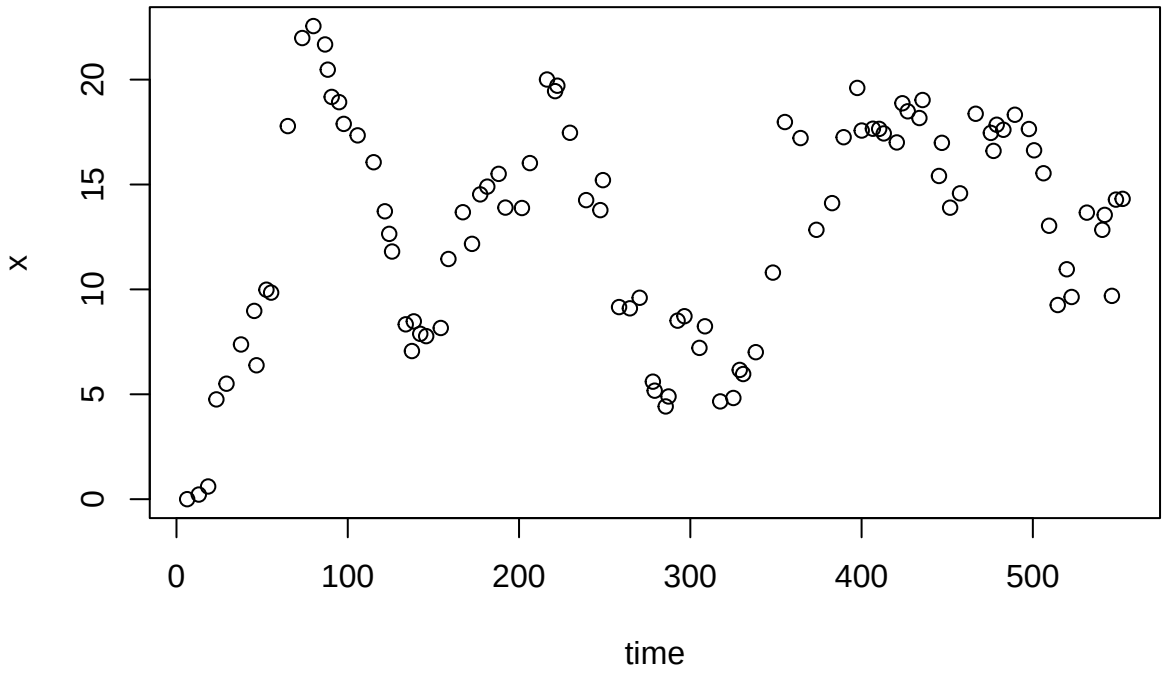

**Fig. S4.** Estimated state-dependent distributions along with 95% pointwise credible intervals for each snake.

## S4.1 Diffusion in 1D

Before constructing two-dimensional spatial HMMs, we will begin by describing the process of constructing these models in the simplest case: simple Brownian motion in one dimension (c.f. Okubo and Levin (2001)).

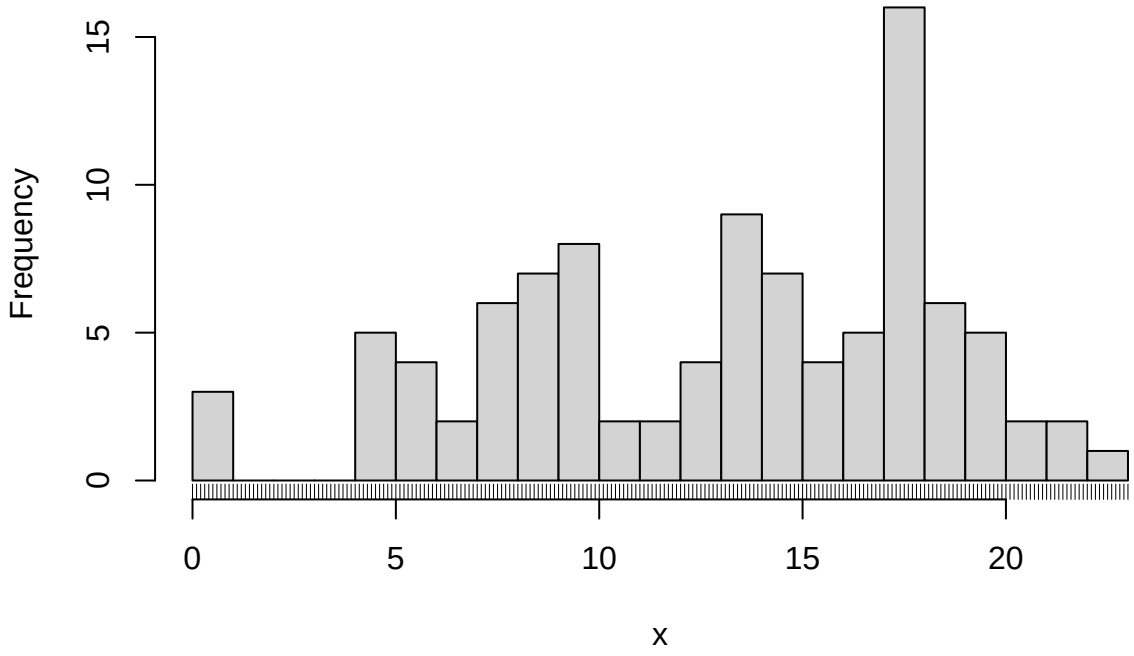
Let's simulate some Brownian motion movement in 1D with irregular time intervals between observations.

```
set.seed(52810)

n <- 100 # number of observations

sd <- 1 # diffusion standard deviation

obst <- cumsum(runif(n, 1, 10)) # observation times

dt <- diff(obst) # time between observations

x <- cumsum(c(0, rnorm(n - 1, 0, sd * sqrt(dt))))

plot(obst, x, xlab = "time")
```



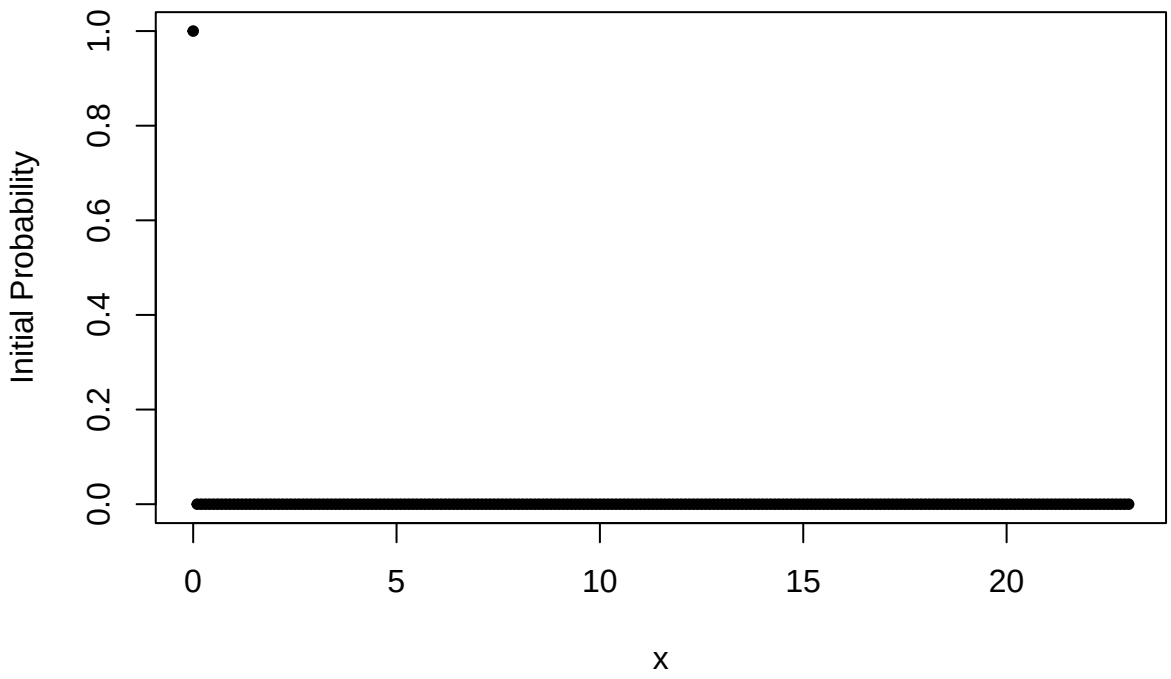

The idea behind spatial HMMs is to discretize the space where movement occurs, so here that is approximately between 0 and 23. Let's create the grid.

```
dx <- 0.1 # grid spacing

g <- seq(0, 23, by = dx) # grid

ng <- length(g) # number of grid cells

hist(x, breaks = 30, main = "")

rug(g)
```




Let's suppose we know the animal begins at position $x = 0$. We can describe this in a vector with a 1 for the grid cell that represents 0 (1 because we know the individual occupies that grid cell with probability 1).

```
p0 <- rep(0, ng)

p0[1] <- 1

plot(g, p0, xlab = "x", ylab = "Initial Probability", pch = 20)
```




The goal when applying spatial HMMs is to predict from this initial probability vector, the probability the animal will be at any other position $x$ some time $t$ in the future. Once we can compute this prediction, we can compare it to what is truly observed and therefore determine the optimal value for the movement parameters.

For Brownian motion, we know that in *continuous* space, *continuous* time, the probability of being at location $x$ after a time $t$, $p(x,t)$, is described by the equation (Okubo and Levin, 2001):

$$\frac{\partial p}{\partial t} = \frac{\sigma^2}{2}\frac{\partial^2 p}{\partial x^2}$$

An HMM requires you to specify the transition rates (*rates* here, not probabilities as we are working in *continuous* time) between grid cells (Zucchini et al., 2017). This equation describes transitions in continuous space, so we must discretize it onto the grid we just created.

One way to do this is to use finite differencing (Quarteroni and Valli, 2008). Inside a grid cell $k$ where $k > 1$ and less than the total number of grid cells, $n_g$, the second derivative can be approximated by:

$$\frac{p_{k+1} - 2p_k + p_{k-1}}{h^2}$$

where $p_i$ is the probability mass in grid cell $i$ and $h$ is the grid spacing. When $k = 1$ or $k = n_g$, we have respectively,

$$\frac{p_{k+1} - p_k}{h^2}, \frac{p_k - p_{k-1}}{h^2}$$

as we can effectively assume the derivative is zero at the boundary of the grid.

This means that for the vector of probability masses over the whole grid, $\mathbf{p}$, the equation becomes

$$\frac{\partial \mathbf{p}}{\partial t} = \frac{\sigma^2}{2}\mathbf{G}\mathbf{p}$$

where $\mathbf{G}$ is a *sparse* matrix with $k^{\text{th}}$ row all zeroes except in positions $(k-1, k, k+1)$ that have entries $(1, -2, 1)/h^2$. The first row has first two entries $(-1, 1)/h^2$ and the last row has last two entries $(1, -1)/h^2$.

We can create this matrix in `R`:

```
G <- bandSparse(ng, ng, k = c(-1, 0, 1),
                diagonals = list(rep(1, ng - 1),
                                 c(-1, rep(-2, ng - 2), - 1),
                                 rep(1, ng - 1)))
```

```
G <- G / dx^2

G[1:10, 1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
##  [1,] -100  100    .    .    .    .    .    .    .    .
##  [2,]  100 -200  100    .    .    .    .    .    .    .
##  [3,]    .  100 -200  100    .    .    .    .    .    .
##  [4,]    .    .  100 -200  100    .    .    .    .    .
##  [5,]    .    .    .  100 -200  100    .    .    .    .
##  [6,]    .    .    .    .  100 -200  100    .    .    .
##  [7,]    .    .    .    .    .  100 -200  100    .    .
##  [8,]    .    .    .    .    .    .  100 -200  100    .
##  [9,]    .    .    .    .    .    .    .  100 -200  100
## [10,]    .    .    .    .    .    .    .    .  100 -200
```

```
G[(ng - 9):ng, (ng - 9):ng]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
##  [1,] -200  100    .    .    .    .    .    .    .    .
##  [2,]  100 -200  100    .    .    .    .    .    .    .
##  [3,]    .  100 -200  100    .    .    .    .    .    .
##  [4,]    .    .  100 -200  100    .    .    .    .    .
```

```
##  [5,]    .    .    .  100 -200  100    .    .    .    .
##  [6,]    .    .    .    .  100 -200  100    .    .    .
##  [7,]    .    .    .    .    .  100 -200  100    .    .
##  [8,]    .    .    .    .    .    .  100 -200  100    .
##  [9,]    .    .    .    .    .    .    .  100 -200  100
## [10,]    .    .    .    .    .    .    .    .  100 -100
```

This matrix differential equation only involves continuous-time. We have removed the continuous space component and replaced it with discrete space in terms of vectors and a matrix. We will not go into detail how the continuous-time, discrete-space equation is solved. It is well known (c.f. Sidje (1998)) in matrix calculus that the solution is given by

$$\mathbf{p}_t = \exp\left(\frac{\sigma^2}{2}\mathbf{G}t\right)\mathbf{p}_0$$

where $\mathbf{p}_t$ is the probability mass vector over the grid at time $t$. So, for example, the $k^{\text{th}}$ entry of $\mathbf{p}_t$ would give the probability the animal is in grid cell $k$ after $t$ time units. Notice that to compute $\mathbf{p}_t$, we must compute the matrix exponential.

In this one-dimensional case, the matrix $\mathbf{G}$ is a $231 \times 231$ matrix. This is not unreasonably large when calculating the matrix exponential. In two-dimensional cases, however, unreasonably large matrices will be easily encountered.

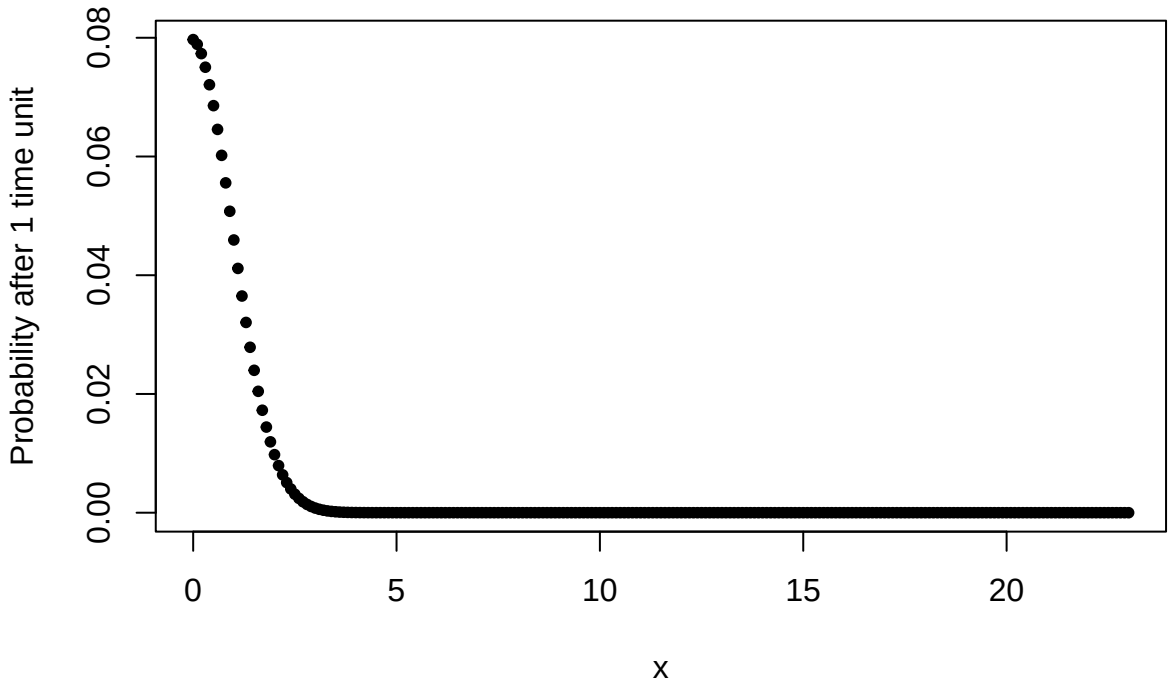
There are many methods to compute the matrix exponential (see the main paper for a discussion of these). In this appendix, we will use the Krylov subspace method (Sidje, 1998) which we have implemented in the `pathIntegrateR` package. This method takes advantage of the fact that the matrix $\mathbf{G}$ is sparse.

To use this method we will convert the sparse matrix $G$ into a row-major matrix as this speeds up computation (since the multiplication above is done by row of the matrix):

```
Q <- sd^2 / 2 * G

Q <- as(Q, "RsparseMatrix") # convert to row-major
```

Now, let's compute $\mathbf{p}_t$ for $t = 1$:

```
pt <- sparse_action(Q, p0, t = 1)

plot(g, pt, xlab = "x", ylab = "Probability after 1 time unit", pch = 20)
```




Notice this method of computing the matrix exponential is still beneficial in this case:

```
# krylov subspace way

system.time(pt <- sparse_action(Q, p0, t = 1))
```

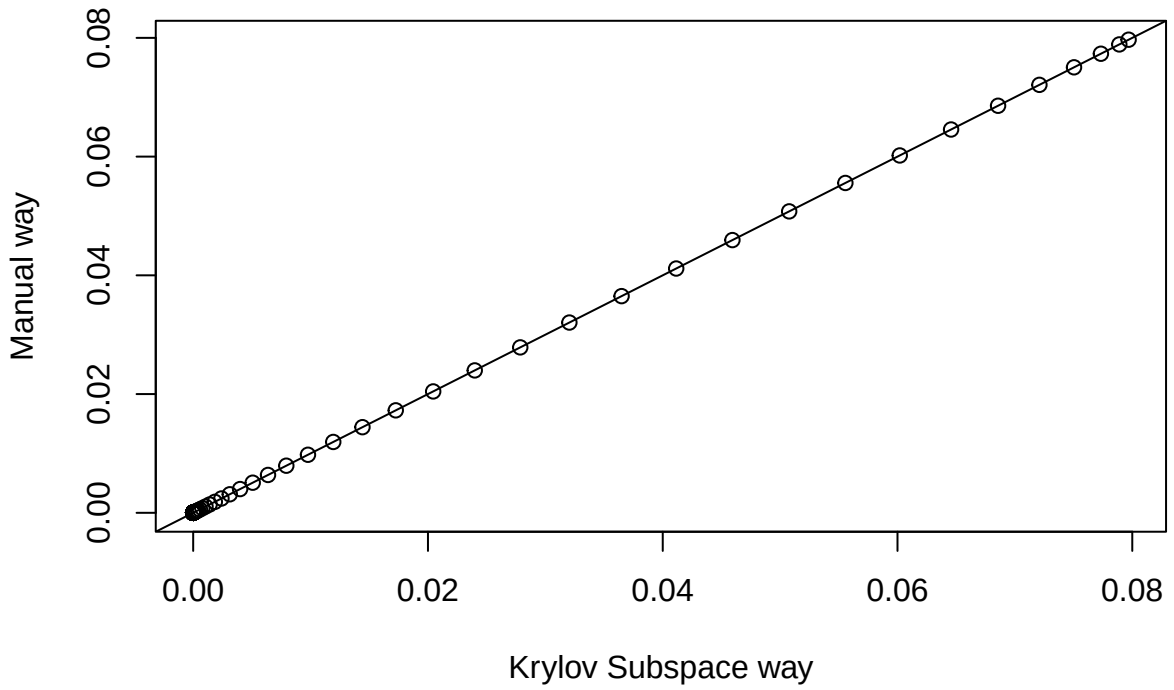
```
##    user  system elapsed
```

```
##    0.001    0.000    0.001
```

```
# manual way

system.time(pt2 <- Matrix::expm(Q * 1) %*% p0)
```

```
##    user  system elapsed
##    0.090    0.006    0.020
```

```
# get the same answer

plot(pt, pt2, xlab = "Krylov Subspace way", ylab = "Manual way")

abline(a = 0, b = 1)
```




We now have a method to compute the probability an individual will be in position $x$ at time $t$ given the animal is in position $x_0$ at time $t_0$. This is all that is needed for a Markov process model. We can then write the standard HMM negative log-likelihood as follows (Zucchini et al., 2017):

```r
#' Compute negative log-likelihood for 1D diffusion spatial HMM
#'
#' @param log_sd log of diffusion standard deviation parameter
#' @param obs grid cells where locations are observed
#' @param obst times of observations
#' @param G second-derivate matrix
#' @param p0 initial probability vector
#'
#' @return negative log-likelihood
calc_diffusion_nllk <- function(log_sd, obs, obst, G, p0) {
  # initial probability
  p <- p0
  nllk <- 0
  # compute diffusion parameter
  sd <- exp(log_sd)
  # create transition rate matrix
  Q <- as(sd^2 / 2 * G, "RsparseMatrix")
  # loop over observations
  # first observation is taken as given and not modelled
  for (i in 2:length(obst)) {
    # update probability mass to new time
    p <- sparse_action(Q, p, t = obst[i] - obst[i - 1])
```

```
    # get probability of new location observed
    tmp <- p[obs[i]]
    # set to zero for locations not observed
    p <- rep(0, length(p0))
    p[obs[i]] <- tmp
    # add to negative log-likelihood
    psum <- sum(p)
    nllk <- nllk - log(psum)
    # rescale p
    p <- p / psum
  }
  return(nllk)
}
```

Let's estimate the diffusion parameter from the data. To make computation faster, we will work out what grid cell each observation lands in.

```
# find out grid cell each observation is in
grid_cell_centers <- g + dx / 2
obs <- sapply(x,  FUN = function(i) {which.min(abs(grid_cell_centers - i))})


# set initial parameter value for optimiser
ini_sd <- log(runif(1, 0.2, 2))
```

```r
# fit model
opt <- nlminb(ini_sd, calc_diffusion_nllk,
              obs = obs, obst = obst, G = G, p0 = p0,
              control = list(trace = 1))
```

```
##   0:     452.24250: 0.396538
##   1:     442.71725: 0.133984
##   2:     441.96067: -0.0870146
##   3:     441.20726: 0.0135727
##   4:     441.19130: 0.00165007
##   5:     441.19116: 0.000430700
##   6:     441.19116: 0.000447534
```

So our final estimate is 1.0004 compared to the true value of 1. We can now treat this like any other HMM: check pseudo-residuals, compute state probabilities, and use the Viterbi algorithm to decode the most likely path (sequence of grid cells) the animal moved through (see Zucchini et al. (2017) for all these details).

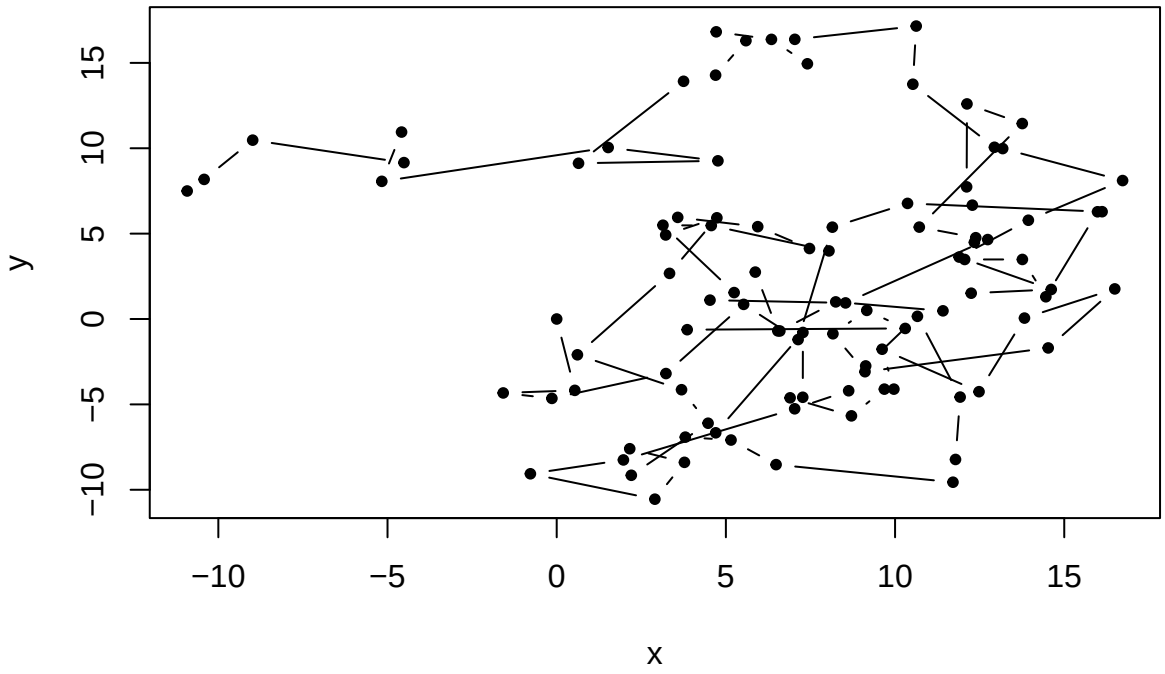
This was a simple example and, of course, for diffusion we can estimate the diffusion parameter without spatial HMMs. The advantage of this approach will be clearer when state-switching is introduced. Nonetheless, two clear advantages of this approach are already apparent: (1) any observation type can be accommodated as with standard HMMs; (2) spatially-varying diffusion parameters naturally can be accommodated, a case where a closed, analytic solution is not readily available.

## S4.2 Diffusion in 2D

The two-dimensional spatial HMM is very similar to the 1D case discussed above. An obvious alternation is the need to specify a 2D grid over space. The most complicated change, however, is in the $\mathbf{G}$ matrix.

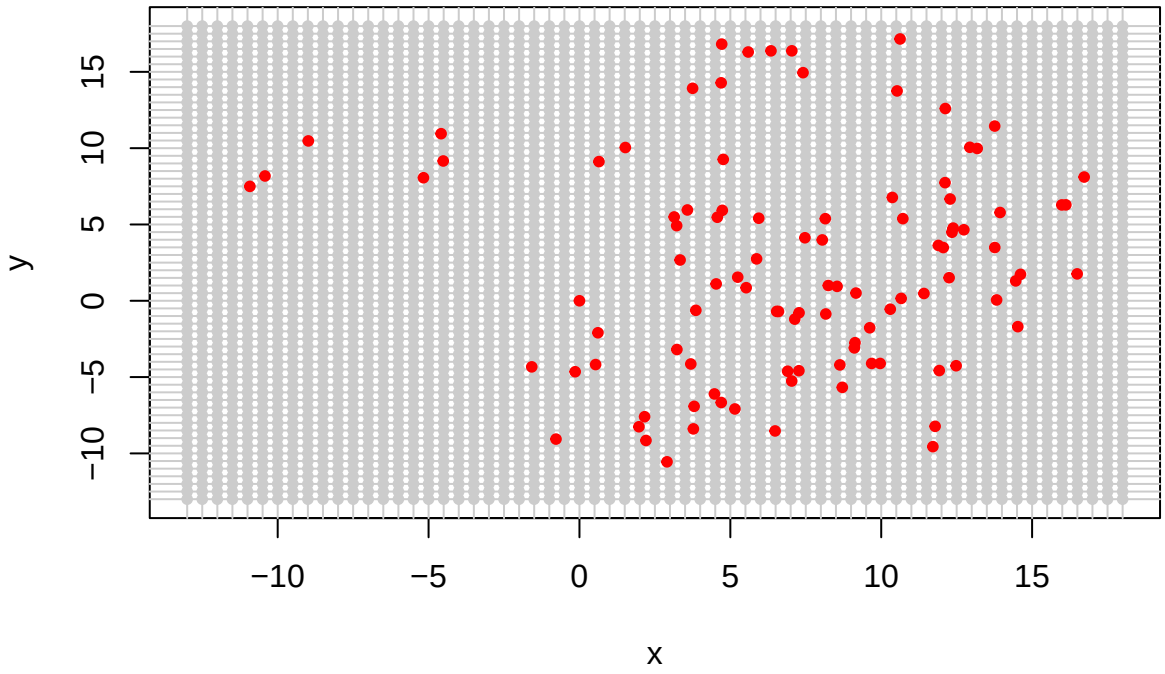
Let's start the same way as in the 1D case and simulate Brownian motion:

```
set.seed(42162)

n <- 100 # number of observations

sd <- 1 # diffusion standard deviation

obst <- cumsum(runif(n, 1, 10)) # observation times

dt <- diff(obst) # time between observations

x <- cumsum(c(0, rnorm(n - 1, 0, sd * sqrt(dt)))) # x-direction

y <- cumsum(c(0, rnorm(n - 1, 0, sd * sqrt(dt)))) # x-direction

plot(x, y, pch = 20, type = "b")
```




The discrete grid can be, in theory, any segmentation of space either into irregular or regular polygons (for example, see Pedersen et al. (2011b)). For simplicity, we will use a regular square grid.

```
dx <- 0.5 # grid spacing in each dimension
g <- seq(-13, 18, by = dx) # grid in 1D
gr <- expand.grid(g, g) # 2D grid
ng <- length(g) # number of grid cells
# plot grid
plot(gr, pch = 20, col = "grey80", xlab = "x", ylab = "y")
abline(v = g, col = "grey80")
abline(h = g, col = "grey80")
points(x, y, pch = 20, col = "red")
```




Now, for the finite difference matrix. Recall, in the 1D case that $\mathbf{G}$ is a *sparse* matrix with $k^{\text{th}}$ row all zeroes except in positions $(k-1, k, k+1)$ that have entries $(1, -2, 1)/h^2$. Intuitively, this is so because in an infinitesimally small time, the individual can either stay in the grid cell they are in, move left, or move right. In the 2D case, by analogy, we would then expect $\mathbf{G}$ to have nine non-zero entries: stay in the cell you are in, move left, move right, move up, move down, or move diagonally in one of four directions. Using the mathematical partial differential equation (as we did above), this is not exactly how it turns out. Instead we have five non-zero entries: stay where you are, move left, move right, move up, and move down. This is because the diagonal cells meet at only a single point that is, theoretically, infinitely small and so no transitions can occur in that direction over an infinitely small time period.

Mathematically, for 2D Brownian motion we have that the probability density at time $t$ in 2D location $(x, y)$ is $p(x, y, t)$ and

$$\frac{\partial p}{\partial t} = \frac{\sigma^2}{2}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right)$$

Now, how can this be translated into a discrete-space, continuous-time equation? Again, one can use finite differencing (Quarteroni and Valli, 2008), the right-hand side derivatives become:

$$\frac{p_L - 2p_C + p_R}{h^2} + \frac{p_U - 2p_C + p_D}{h^2}$$

where $L, C, R, U, D$ reference the cell to the left, centre, right, up, and, down relative to the centre cell.

An efficient (and for more complicated models easier) way to specify the matrix $\mathbf{G}$ is by using *Kronecker products*.

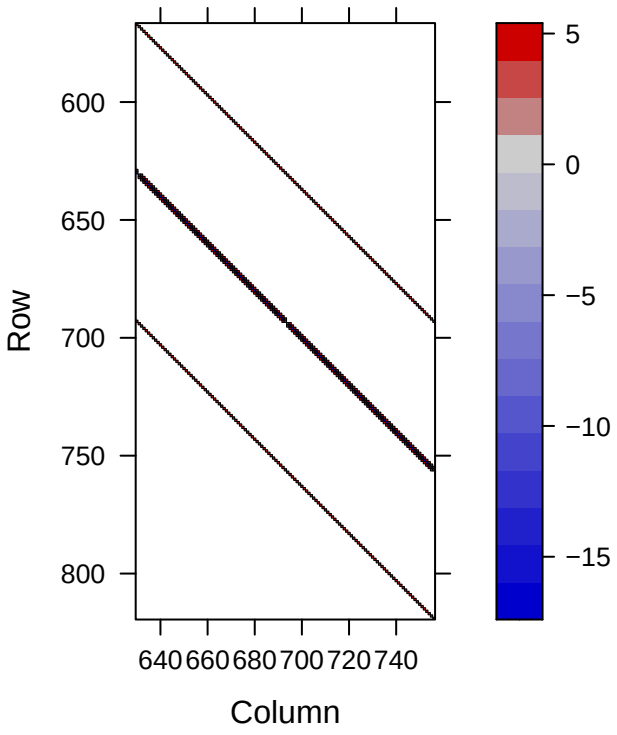
```
# number of grid points in x direction and total
Nx <- Ny <- length(g)
N <- Nx * Ny


# compute derivative matrices in 1D
Gx <- bandSparse(Nx, Nx, k = c(-1, 0, 1),
                 diagonals = list(rep(1, Nx - 1),
                                  rep(-2, Nx),
                                  rep(1, Nx - 1)))
Gx[1, 1] <- -1
Gx[Nx, Nx] <- -1
```

```
Gy <- bandSparse(Ny, Ny, k = c(-1, 0, 1),
                 diagonals = list(rep(1, Ny - 1),
                                  rep(-2, Ny),
                                  rep(1, Ny - 1)))
Gy[1, 1] <- -1
Gy[Ny, Ny] <- -1


# make these into 2D operators
Ix <- diag(Nx)
Iy <- diag(Ny)
G <- (Iy %x% Gx) + (Gy %x% Ix)
G <- G / dx^2
```

The matrix **G**, for each row (ignoring boundary cells), has five non-zero entries. The rows are ordered so that they describe the movement from each cell in order of y-value first and x-value second, i.e. the order of the rows corresponds to tracing your finger over the 2D spatial grid by row (along the x-axis first).

We can look at the structure of this matrix:

```
image(G, xlim = c(Nx*10, Nx*12), ylim = c(Nx*9, Nx*13))
```

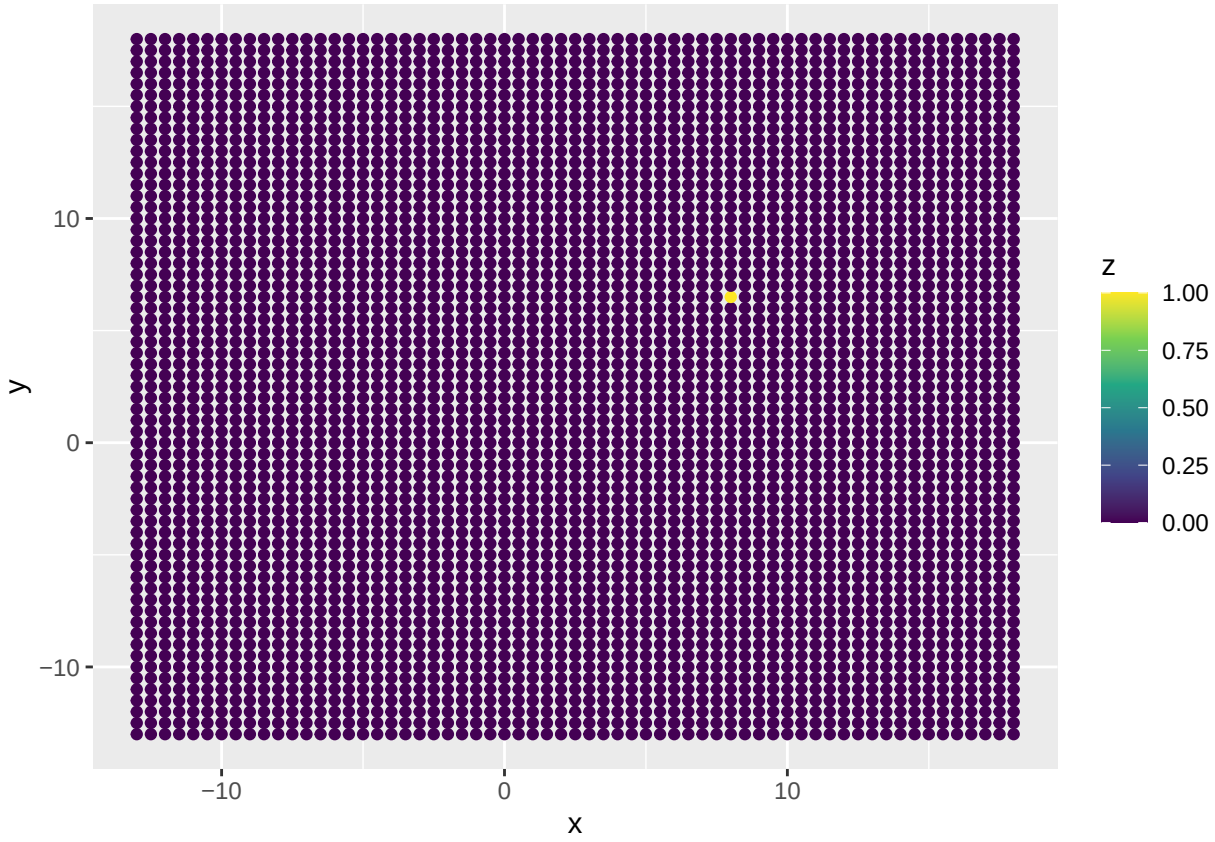



What we can see is that for every grid cell (row) there is a chance of staying in that grid cell (the diagonal), a chance of moving left or right (the cells immediately around the diagonal) and a chance of moving up or down (the cells far to the left and right of the diagonal).

We can now do an example of a single step of Brownian motion using the Krylov subspace method. The solution to the 2D partial differential equation once in discrete-space is the same as the 1D case.

```
# initial distribution

p0 <- rep(0, N)

p0[2500] <- 1
```

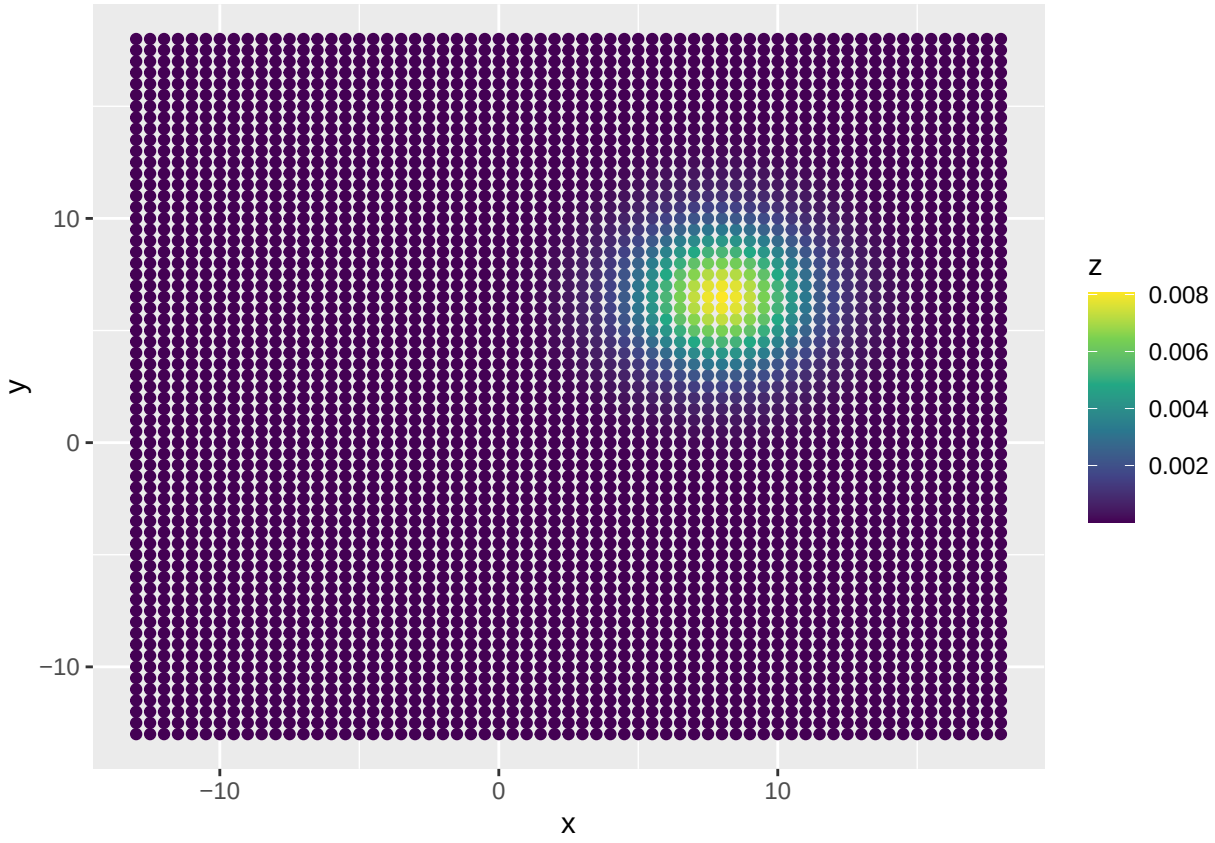
```
ggplot(data.frame(x = gr[,1], y = gr[,2], z = p0)) +
  geom_point(aes(x = x, y = y, col = z)) +
  scale_color_viridis_c()
```




Notice in this case that we have nearly a $4000 \times 4000$ matrix to compute the matrix exponential of. This is a more difficult computation and where the Krylov method will reap more benefits.

```
Q <- sd^2 * G / 2

Q <- as(Q, "RsparseMatrix")

p <- pathintegrateR::sparse_action(Q, p0, t = 5)


ggplot(data.frame(x = gr[,1], y = gr[,2], z = p)) +
```

```
  geom_point(aes(x = x, y = y, col = z)) +

  scale_color_viridis_c()
```




As in the 1D case, we can now estimate the diffusion parameter. In fact, we can use the same likelihood function. We must first determine what grid cell each observation occurs within.

```
# find out grid cell each observation is in

cell_centresx <- g + dx / 2

xobs <- sapply(x, FUN = function(x) {which.min(abs(x - cell_centresx))})

cell_centresy <- g + dx / 2

yobs <- sapply(y, FUN = function(x) {which.min(abs(x - cell_centresy))})

obs <- xobs + (yobs - 1) * Nx
```

```r
# set initial parameter value for optimiser
ini_sd <- log(runif(1, 0.2, 2))


# fit model
opt <- nlminb(ini_sd, calc_diffusion_nllk,
              obs = obs, obst = obst, G = G, p0 = p0,
              control = list(trace = 1))
```

```
##   0:     629.08525: 0.664719
##   1:     620.83102: -0.335281
##   2:     586.04078: 0.164719
##   3:     584.47011: 0.0944038
##   4:     584.30258: 0.0603501
##   5:     584.30075: 0.0636157
##   6:     584.30075: 0.0635239
##   7:     584.30075: 0.0635230
```

The final estimate is 1.0656 and the true value is 1.

### S4.3 State-switching

State-switching is were spatial HMMs are most useful (Pedersen et al., 2011a). This is because using the above framework we can allow for continuous-time state-switching without needing to

assume that switches occur at specific times only and without needing to discretise the movement model in time.

The adaptation of 2D to state-switching 2D is similar to the adaptation from 1D to 2D. By adding a dimension (1D to 2D) we added non-zero elements to the matrix $\mathbf{G}$ to capture the intuitive idea of where an individual can move in 2D over a infinitely small time interval. State-switching is just another dimension. The difference is that instead of adding a new spatial dimension (which is discretised into hundreds of cells), we add a behaviour state, intuitively a behaviour space that is discretised, typically, into two or three cells. The idea is that the individual moves in a different way depending on where it is in behaviour space.

As with the adaptation to 2D, adding state-switching involves adding new non-zero elements to the $\mathbf{G}$ matrix and expanding the space we are discretising from a 2D grid to a 3D grid. Each cell in the 3D grid represents a 2D location and a particular behaviour. You can imagine a series of 2D grids stacked on top of each other. The animal can now move not only only in 2D space (along these horizontal grids), but can also change behaviour (move up and down the stack). Depending on its behaviour, the animal will move differently in 2D space.

Mathematically, this is described by adding more terms to the partial differential equation (Pedersen et al., 2011a). Let $p_b(x, y, t)$ be the probability density that an animal is in location $(x, y)$ at time $t$ in behaviour $b$. Further, suppose there are $B$ behaviours in total (typically $B = 2, 3$ or 4). The equation is now

$$\frac{\partial p_b}{\partial t} = \frac{\sigma_b^2}{2}\left(\frac{\partial^2 p_b}{\partial x^2} + \frac{\partial^2 p_b}{\partial y^2}\right) + \sum_{s=1}^{B} \gamma_{s,b} p_s$$

The first part is the 2D Brownian motion as before, it only affects what happens within each behaviour. The final term describes the movement between behaviours: $\gamma_{s,b}$ is a switching rate from state $s$ to state $b$ and by definition $\gamma_{b,b} = -\sum_{s \neq b} \gamma_{b,s}$ . Intuitively, $\gamma_{s,b}$ controls how much of the probability from state $s$ flows to state $b$ (i.e. how likely it is the animal will change behaviour from state $s$ to state $b$).

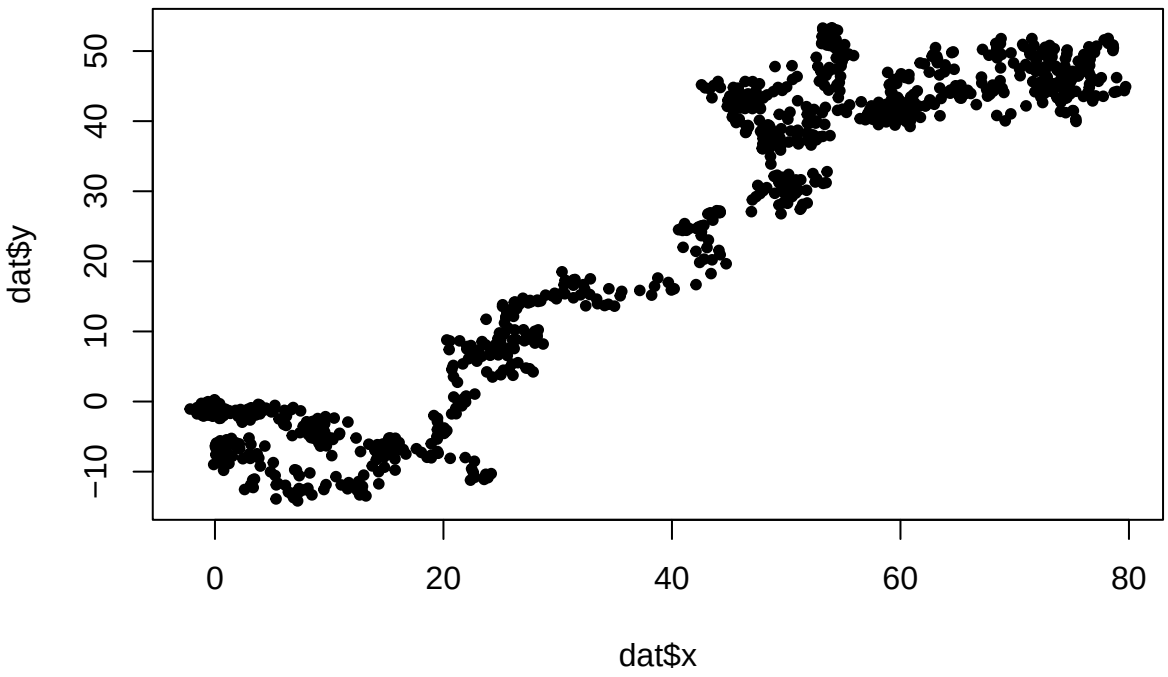
Ultimately, for $B$ behaviours, every row of $\mathbf{G}$ has $B-1$ additional non-zero entries, representing the chance an animal will switch behaviour. In this approximation, animals do not switch behaviour and move in an infinitely small amount of time: they do one or the other. If you think of this as 3D movement, this is for the same reason as why diagonal movement was not included in the 2D case.

Let's simulate some state-switching Brownian motion with two behaviours.

```
# true diffusions
sd <- c(0.5, 1.0)


# behaviour switching
mean_duration <- c(15, 30)
rates <- 1 / mean_duration
trm <- diag(-rates)
trm[!diag(2)] <- rates
trm <- t(trm)


# simulate movement
```

```r
obst <- 1:1000

x <- 0

y <- 0

now <- 0

dt <- 0.1

tpm <- expm(trm * dt)

curt <- 0

cur <- 1

b <- 1

bs <- b

dat <- data.frame(x = 0, y = 0, t = 0)

while (now < max(obst)) {

  x <- x + rnorm(1, 0, sd[b] * sqrt(dt))

  y <- y + rnorm(1, 0, sd[b] * sqrt(dt))

  b <- sample(1:nrow(tpm), size = 1, prob = tpm[b,])

  bs <- c(bs, b)

  while (now + dt > curt & cur < length(obst) + 1) {

    dat <- rbind(dat, c(x, y, now + dt))

    curt <- obst[cur]

    cur <- cur + 1

  }

  now <- now + dt
```

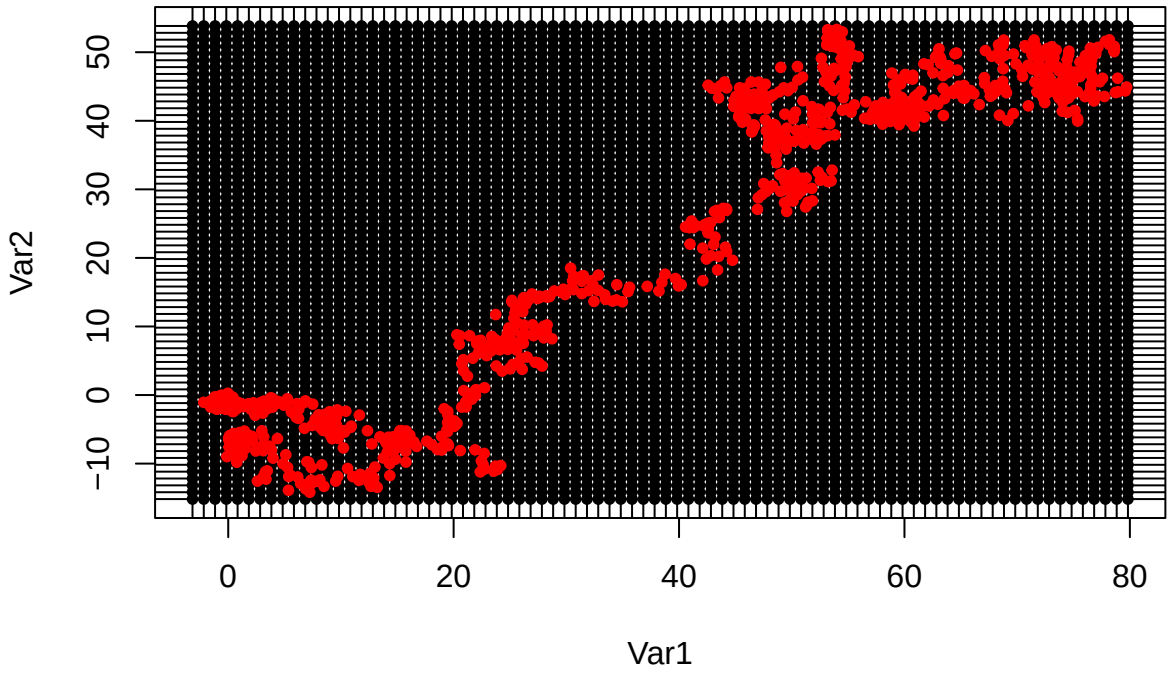
```
}

dat <- dat[-1,]



# plot simulated data

plot(dat$x, dat$y, pch = 20, type = "b")
```




The setup of the grid and the diffusion matrix for 2D is the same as above:

```
# set grid spacing

dx <- 1



# set boundary of space

xrange <- c(min(dat$x) - dx, max(dat$x) + dx)

yrange <- c(min(dat$y) - dx, max(dat$y) + dx)
```

```r
# plot grid

grx <- seq(xrange[1], xrange[2], by = dx)

gry <- seq(yrange[1], yrange[2], by =dx)

gr <- expand.grid(grx, gry)

plot(gr, pch = 20)

abline(v = grx)

abline(h = gry)

points(dat$x, dat$y, pch = 20, col = "red")
```




```r
# number of grid points in x direction and total

Nx <- length(grx)

Ny <- length(gry)

N <- Nx * Ny
```

```
# compute derivative matrices in 1D
Gx <- bandSparse(Nx, Nx, k = c(-1, 0, 1),
                 diagonals = list(rep(1, Nx - 1),
                                  rep(-2, Nx),
                                  rep(1, Nx - 1)))
Gx[1, 1] <- -1
Gx[Nx, Nx] <- -1
Gy <- bandSparse(Ny, Ny, k = c(-1, 0, 1),
                 diagonals = list(rep(1, Ny - 1),
                                  rep(-2, Ny),
                                  rep(1, Ny - 1)))
Gy[1, 1] <- -1
Gy[Ny, Ny] <- -1


# make these into 2D operators
Ix <- diag(Nx)
Iy <- diag(Ny)
G <- (Iy %x% Gx) + (Gy %x% Ix)
G <- G / dx^2
```

Rather than add the state-switching to $\mathbf{G}$ directly, we will create a matrix for the movement in 2D and another matrix for the state-switching. This means all I want to do for $\mathbf{G}$ is have two copies, representing 2D movement in each behaviour state. Again, I use *Kronecker products*.

```
# make into behavior-switching operators

Ib <- diag(2)

Gb <- Ib %x% G
```

The state-switching matrix is dependent on the parameter values and so I will compute it when needed rather than in advance (as I have did with $\mathbf{G}$).

Let's compute a single update. First, I will compute the state-switching matrix for this update using the true transition rate matrix for the behaviour-switching:

```
T <- t(trm) %x% Diagonal(N)
```

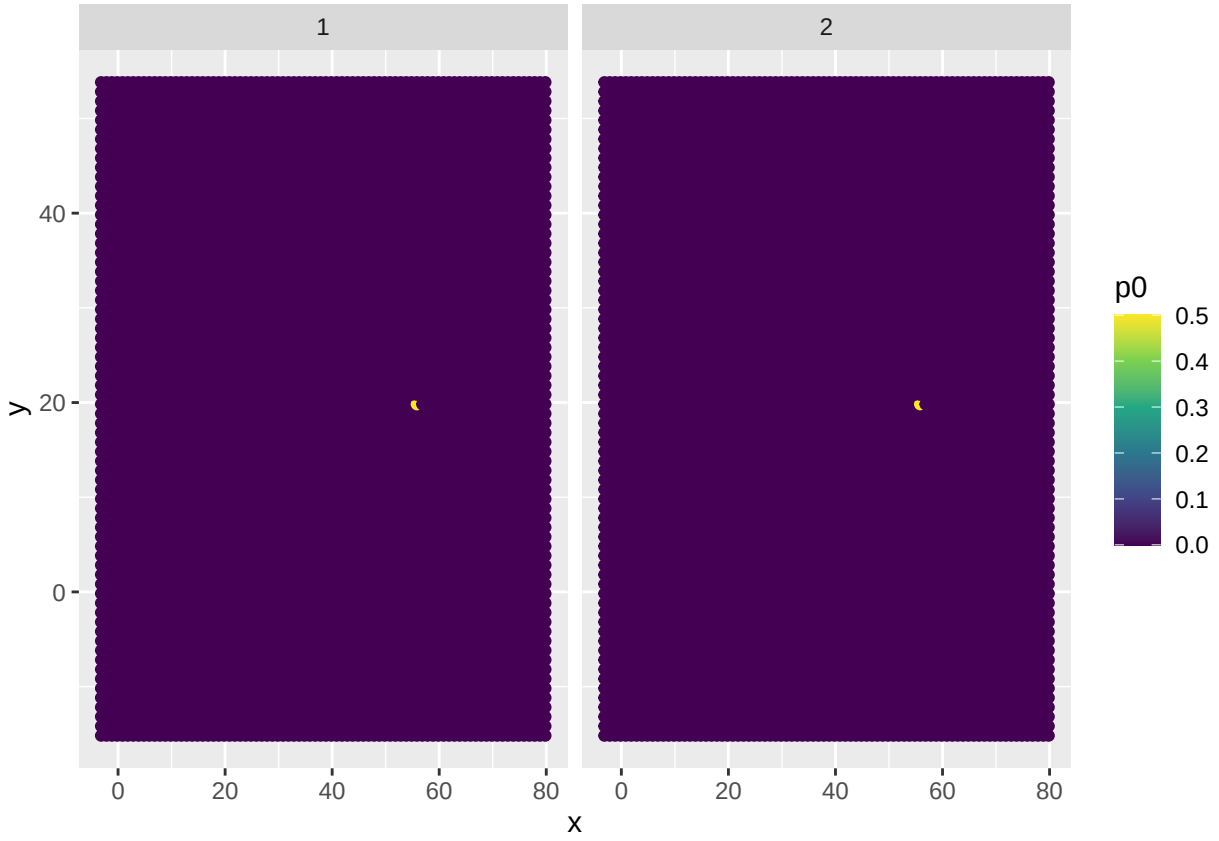
Next, I need to multiply by `sd`. Yet, unlike in previous examples, there are two values of the Brownian motion standard deviation as it depends what behaviour the animal is in. Again, it is important to understand how the grid cells (in 3D space) relate to the rows of $\mathbf{G}$. In the 2D case they were ordered in terms of $y$ first and $x$ last. In the state-switching case, they are ordered behaviour first, $y$ next, and $x$ last. So, to compute the correct movement matrix, I need to multiply the top-left $N \times N$ block of $\mathbf{G}$ by `sd[1]` (where $N$ is the total number of cells in 2D space) and the bottom-right $N \times N$ block by `sd[2]`.

```
# compute diagonal to multiply rows of G

D <- Diagonal(2 * N, x = c(rep(sd[1], N), rep(sd[2], N)))
```

```r
# compute update matrix: movement + behaviour switching

Q <- Gb %*% D / 2 + T

Q <- drop0(Q) # removes unnecessary zeroes

Q <- as(Q, "RsparseMatrix") # make row-major form
```

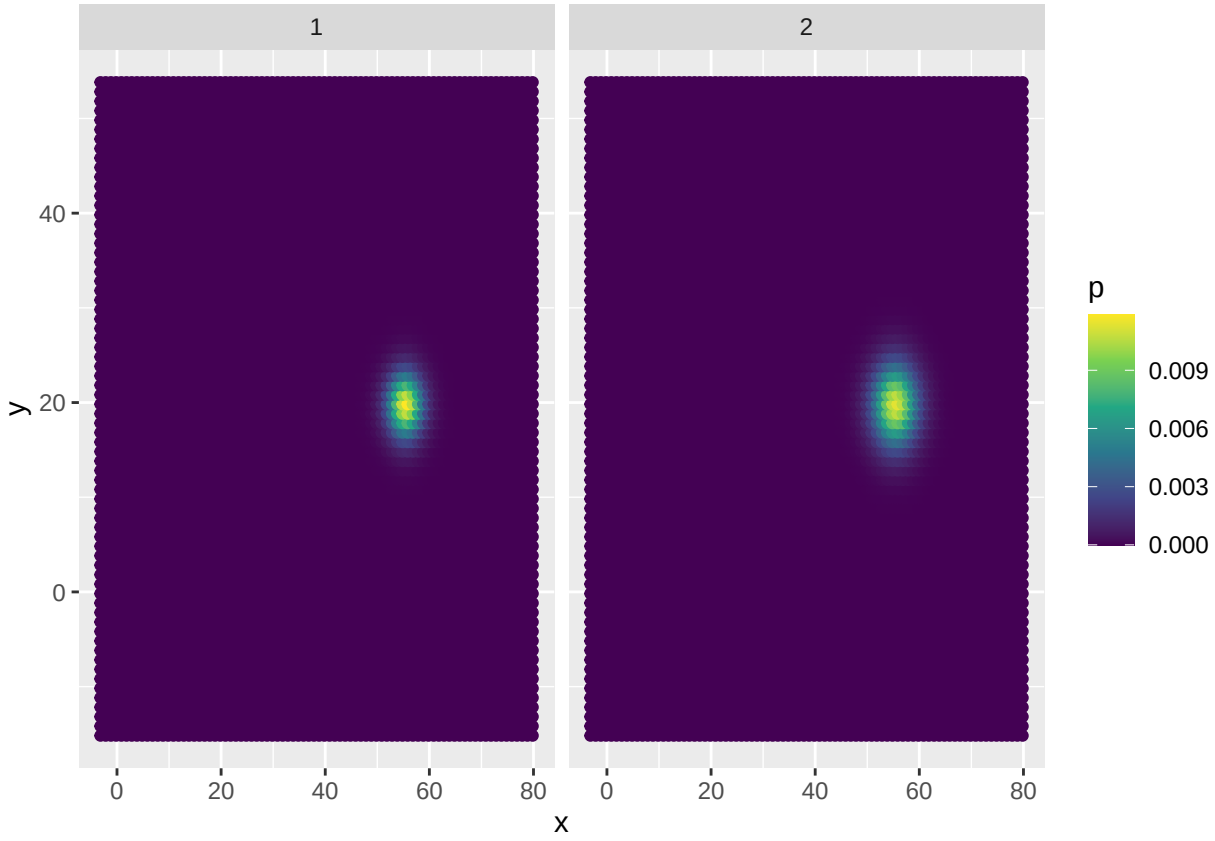
Let's specify an initial probability vector in this 3D space and then update it.

```r
# initial distribution

p0 <- rep(0, N * 2)

p0[3000] <- 0.5 # equally likely to be in each behaviour

p0[3000 + N] <- 0.5


tmp <- data.frame(x = rep(gr[,1], 2),

                  y = rep(gr[,2], 2),

                  state = rep(1:2, each = nrow(gr)),

                  p0 = p0)

ggplot(tmp) +

  geom_point(aes(x = x, y = y, col = p0)) +

  facet_wrap(~state) +

  scale_color_viridis_c()
```




In this case we are taking the matrix exponential of a sparse matrix with approximately 12000 rows and columns.

```
p <- pathintegrateR::sparse_action(Q, p0, t = 10)



tmp$p <- p

ggplot(tmp) +

  geom_point(aes(x = x, y = y, col = p)) +

  facet_wrap(~state) +

  scale_color_viridis_c()
```




You will notice in behaviour 1 that the animal is predicted to diffuse more slowly, as expected. This graphic, however, only shows where the animal would be; we can also see this picture in behaviour space only.

```
pb <- c(sum(p[1:N]), sum(p[(N + 1): (2 * N)]))

pb
```

```
## [1] 0.394646573523 0.605353426477
```

```
# manual way to compute this

c(0.5, 0.5) %*% expm(trm * 10)
```

```
## 1 x 2 Matrix of class "dgeMatrix"
```

```
##                 [,1]            [,2]
## [1,] 0.394646573529 0.605353426471
```

Thus, not only are we solving for where the animal is in space, but also allowing for the animal switching behaviour *at any time* in between observations.

Now, to fit the model. First, we compute what grid cells in 2D space the animal was observed in.

```
# work out cell for each observation
cell_centresx <- grx + dx / 2
xobs <- sapply(dat$x, FUN = function(x) {which.min(abs(x - cell_centresx))})
cell_centresy <- gry + dx / 2
yobs <- sapply(dat$y, FUN = function(x) {which.min(abs(x - cell_centresy))})
obs <- xobs + (yobs - 1) * Nx
```

The likelihood function is a little different for state-switching as the animal, when observed, could occupy either behaviour (i.e. only location in 2D space is observed).

```
calc_ss_nllk <- function(par, obs, obst, p0, Gb, N) {
  llk <- 0
  p <- p0
  D <- Diagonal(2 * N, x = c(rep(exp(par[1]), N), rep(exp(par[2]), N)))
  trm <- matrix(0, nr = 2, nc = 2)
  trm[!diag(2)] <- exp(par[3:4])
  diag(trm) <- -rowSums(trm)
```

```
  T <- t(trm) %x% Diagonal(N)
  Q <- as(Gb %*% D / 2 + T, "RsparseMatrix")
  for (i in 2:length(obst)) {
    p <- pathintegrateR::sparse_action(Q, p, t = obst[i] - obst[i - 1])
    tmp <- c(p[obs[i]], p[obs[i] + N])
    p <- rep(0, length(p))
    p[obs[i]] <- tmp[1]
    p[obs[i] + N] <- tmp[2]
    psum <- sum(p)
    llk <- llk + log(psum)
    p <- p / psum
  }
  return(-llk)
}
```

Let's fit the model. Recall, the parameters we have here are the standard deviation of the Brownian motion under each state (2 parameters) and the switching rates between these two states (which are equal to the reciprocal of the mean time spent in each state, c.f. *continuous time Markov chains*).

```
# set initial distribution
p0 <- rep(0, 2 * N)
p0[obs[1]] <- 0.5
p0[obs[1] + N] <- 0.5
```

```r
# starting values for optimiser
inipar <- log(c(0.2, 1.5, 1 / 10, 1 / 10))


# fit model
opt <- nlminb(inipar,
              calc_ss_nllk,
              obs = obs,
              obst = obst,
              p0 = p0,
              G = Gb,
              N = N,
              control = list(trace = 1))
```

```
##   0:     2698.9345: -1.60944 0.405465 -2.30259 -2.30259
##   1:     2680.2431: -1.49537 0.144582 -2.35726 -2.25958
##   2:     2670.5308: -1.31284 0.234126 -2.54410 -2.16126
##   3:     2666.7671: -1.18612 0.118571 -2.61042 -2.12413
##   4:     2664.2541: -1.08605 0.262477 -2.67341 -2.10186
##   5:     2661.0265: -0.746487 0.203915 -2.63173 -2.24426
##   6:     2659.3653: -0.775358 0.230482 -2.87443 -2.52765
##   7:     2659.3367: -0.772096 0.265733 -2.97473 -2.52182
##   8:     2658.8810: -0.805632 0.216330 -3.06071 -2.54155
```

```
##   9:     2658.7652: -0.809061 0.221838 -3.15366 -2.59317
##  10:     2658.7367: -0.786582 0.216929 -3.16966 -2.63457
##  11:     2658.7133: -0.803886 0.207413 -3.21495 -2.64217
##  12:     2658.7062: -0.802806 0.212270 -3.25359 -2.66687
##  13:     2658.7052: -0.803220 0.210681 -3.24537 -2.65844
##  14:     2658.7052: -0.803225 0.210702 -3.24515 -2.65882
##  15:     2658.7052: -0.803224 0.210702 -3.24518 -2.65878
```

The estimated and true values are shown below:

```
# Estimated
est <- round(c(exp(opt$par[1:2]), exp(-rev(opt$par[3:4]))), 2)
# True
true <- c(sd, -1/diag(trm))
# Compare
cbind(est, true)
```

```
##        est true
## [1,]  0.45  0.5
## [2,]  1.23  1.0
## [3,] 14.28 15.0
## [4,] 25.67 30.0
```

## S4.4 Advection

Thus far we have considered diffusion motion only. Animals typically move *preferentially* in a given direction, driven by environmental conditions (Preisler et al., 2004). Advection is a term used to refer to this tendency in partial differential equation methods. In an advection-diffusion motion, animals diffuse (as we have discussed above) but they also drift and this drift is biased toward (or away, depending on the direction of preference) from environmental features.

There are, however, current limitations on how advection can be used in this context. To highlight these limitations, we will restrict ourselves to 1D advection. The partial differential equation to describe advection is given by

$$\frac{\partial p}{\partial t} = -v\frac{\partial p}{\partial x}$$

where $v$ is the velocity in the positive $x$ direction and $p(x,t)$ is the probability at location $x$ at time $t$. The solution to this equation is known, you simply take the initial shape $p(x,0)$ and shift it $vt$ units to the right for $v > 0$.

The key idea is the advection speed $v$ can change over space and that advection can be combined with diffusion (Pedersen et al., 2011a). This would allow us to build a 2D movement model where animals move in biased random walks where the preferential direction of movement may depend on environmental covariates that affect $v$.

The advection equation can be discretised by finite differencing (Quarteroni and Valli, 2008). There are two popular ways to do this: central differencing and forward differencing. Both have *severe* drawbacks when implemented.

For central differencing in 1D, the first derivative for grid cell $i$ is approximated by $(p_{i+1} - p_{i-1})/(2h)$. For forward differencing where $v > 0$, the derivative is approximated by $(p_{i+1} - p_i)/h$.

Let's build a grid in 1D and the associated differencing matrices:

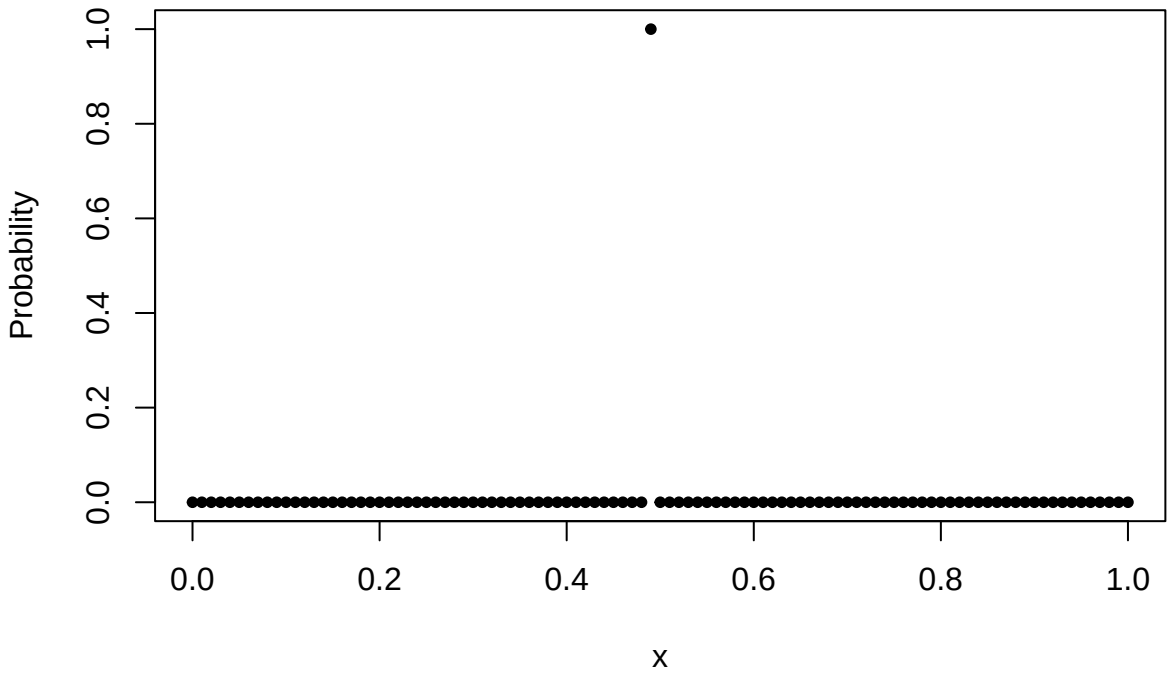
```
dx <- 0.01

gr <- seq(0, 1, 0.01)

N <- length(gr)


# central difference

D <- bandSparse(N, N, k = c(-1, 1),

                diagonals = list(rep(-1, N - 1), rep(1, N - 1)))

D <- D / (2 * dx)


# forward difference (v > 0)

Df <- bandSparse(N, N, k = c(0, -1),

                 diagonals = list(rep(1, N), rep(-1, N - 1)))

Df[N, N] <- 0

Df <- as(Df, "dgCMatrix") / dx
```

Now, we can observe the approximate solutions using the Krylov subspace method.

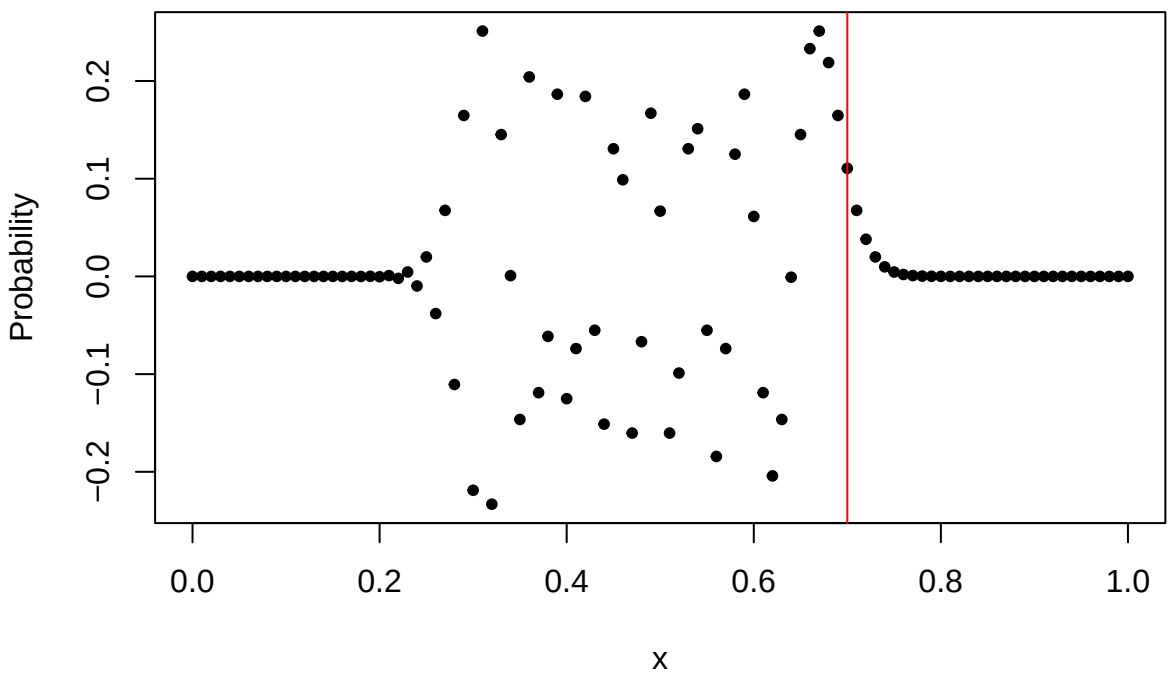
```r
# initial condition

p0 <- rep(0, N)

p0[50] <- 1


# set velocity

v <- 0.2


# set Q matrix for central

Q <- as(-v*D, "RsparseMatrix")


# set Q matrix for forward

Qf <- as(-v*Df, "RsparseMatrix")


# update using central

p <- sparse_action(Q, p0, t = 1)


# update using forward

pf <- sparse_action(Qf, p0, t = 1)
```

Let's consider the central difference solution. Recall, as the initial condition specifies the animal starts exactly at $x = 0.5$ and we know $v = 0.2$, we already know the correct solution is the animal is exactly at $x = 0.7$.

```r
# plot initial condition

plot(gr, p0, pch = 20, xlab = "x", ylab = "Probability")
```

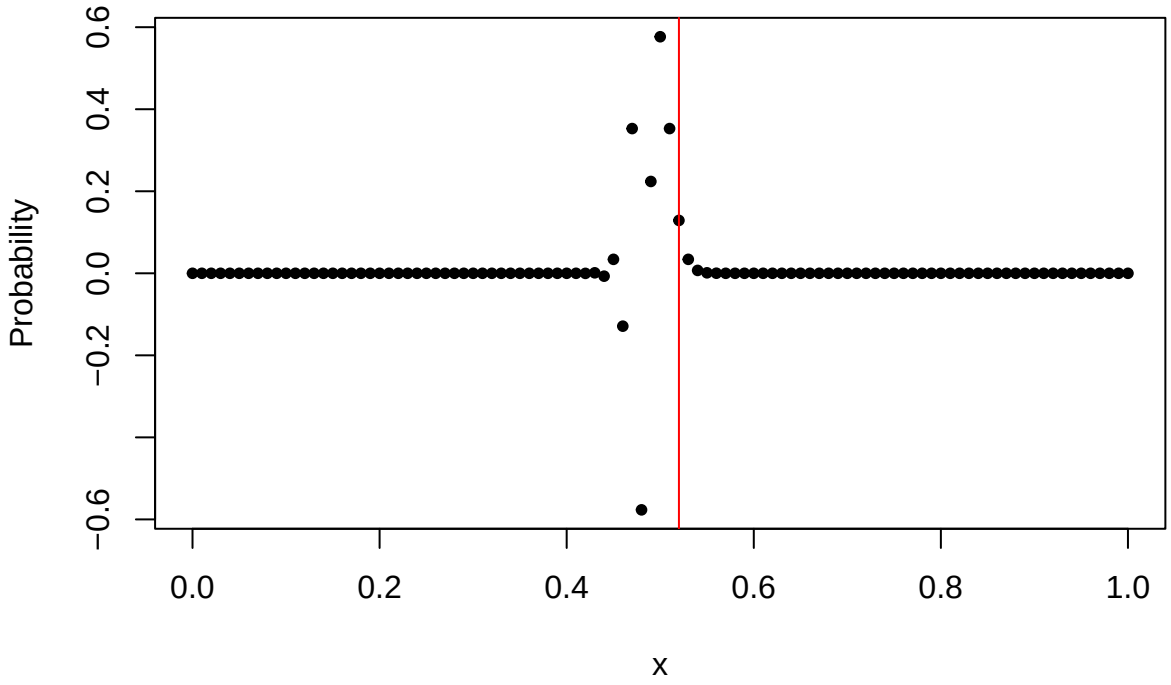



```r
# plot central solution

plot(gr, p, pch = 20, xlab = "x", ylab = "Probability")

abline(v = 0.7, col = "red")
```




We can see that the central difference solution is nonsense. The reason behind this is because of small propagated errors that accumulate as you progress this approximation through time. The approximate solution can even become negative and highly oscillatory. If we compute the central difference solution over a small time period we will see the error is smaller:

```
plot(gr, sparse_action(Q, p0, t = 0.1),
     pch = 20, xlab = "x", ylab = "Probability")
abline(v = 0.5 + 0.2 * 0.1, col = "red")
```

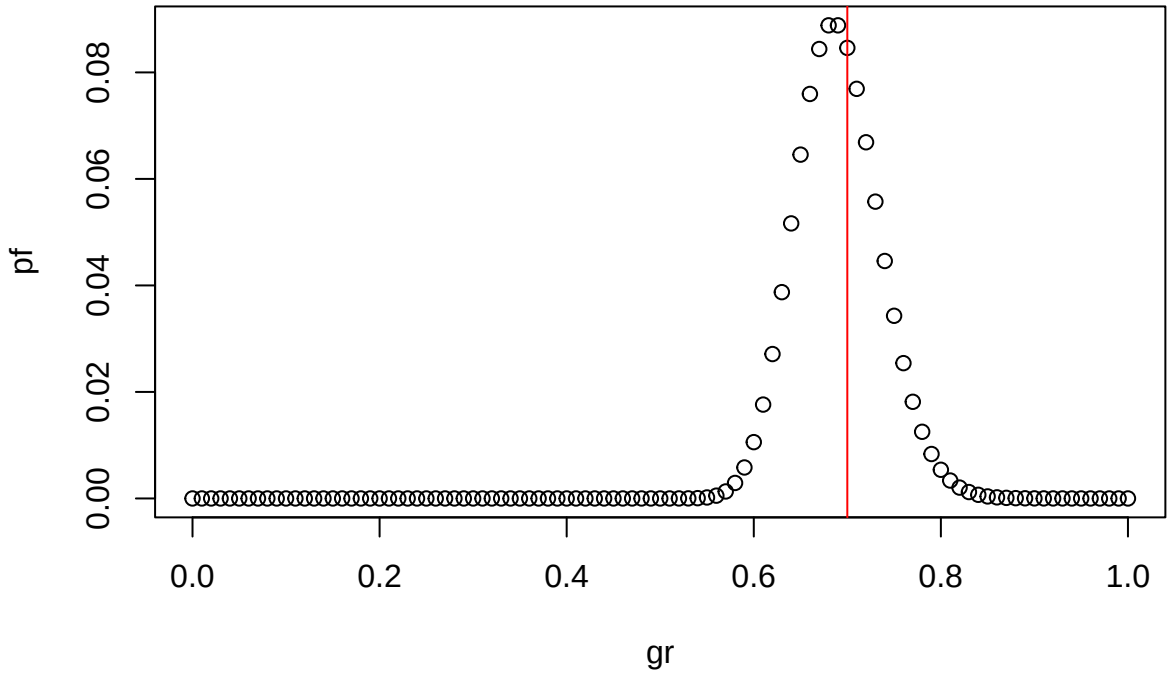



This issue with the central difference operator is well-known. The problem is less acute when the initial condition is more smooth or when diffusion is included in the movement and largely dominates the advection component. Nonetheless, this underlying problem remains and means when advection is large relative to diffusion or initial conditions are peaked (because of good information on animal location, for example) then the likelihood calculation could become nonsensical (i.e. negative values) or high oscillatory (i.e. unrealistic animal movement predictions).

There are methods to prevent these oscillations called *limiters*. In simple terms, these limiters stop the solution from rapidly changing over space, thereby preventing oscillations. However, the downside of using limiters is that you must solve the equation in discrete time. There is no problem with this in theory, but does burden the problem with further computational complexity.

An alternative is the forward difference. It has a different drawback.

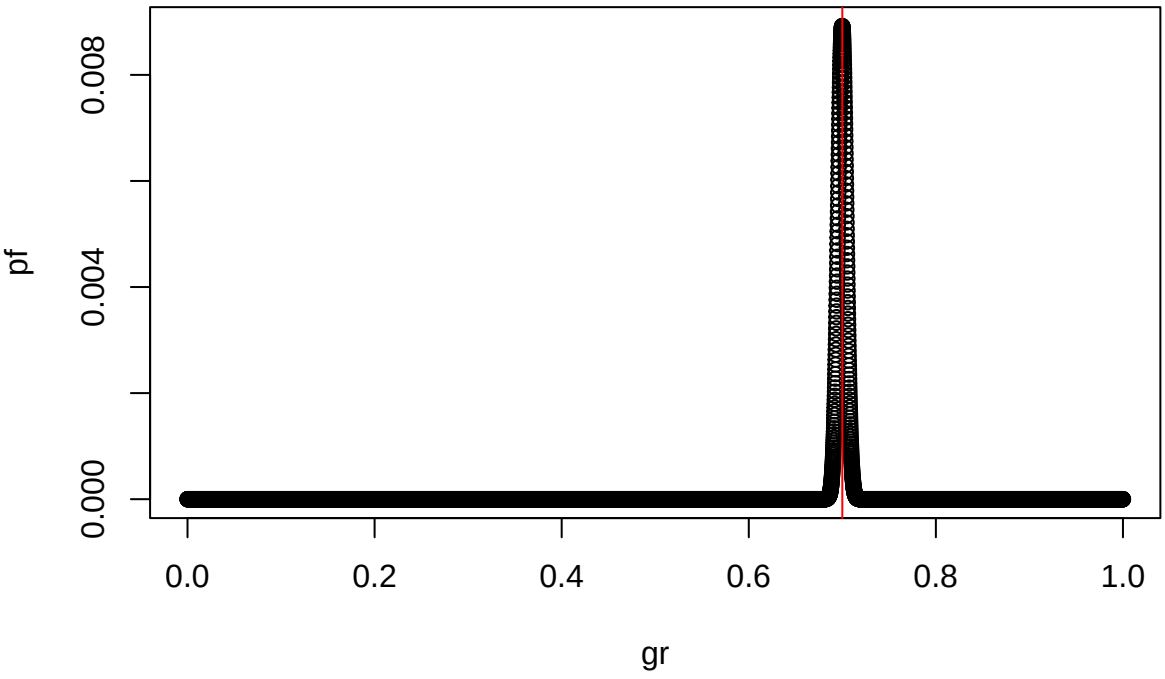
```r
# plot forward solution

plot(gr, pf)

abline(v = 0.7, col = "red")
```




You will notice that this solution is strictly positive and smooth. It is almost centred on the right value of $x = 0.7$. In many respects this approximation is a good one. The drawback is that in reality the solution should be a single point, not a bell curve: there is some diffusion in this approximation despite the fact the continuous-space model is advection only.

This is known as *artificial* diffusion or *numerical* diffusion as it is introduced by the choice of approximation to the advection. The amount of numerical diffusion that occurs depends on the spacing of the grid. For example, here is the forward difference solution with a very small grid spacing:

```r
dx <- 0.0001

gr <- seq(0, 1, dx)

N <- length(gr)

Df <- bandSparse(N, N, k = c(0, -1),

                 diagonals = list(rep(1, N), rep(-1, N - 1)))

Df[N, N] <- 0

Df <- as(Df, "dgCMatrix") / dx

p0 <- rep(0, N)

p0[5000] <- 1

Qf <- as(-v*Df, "RsparseMatrix")

pf <- sparse_action(Qf, p0, t = 1)



plot(gr, pf)

abline(v = 0.7, col = "red")
```




The problem with the forward difference is that the amount of numerical diffusion introduced depends on the grid **and** the value of $v$. If this advection model were combined with a diffusion model, the estimated diffusion standard deviation will be *negatively biased* compared to the true diffusion as some of the diffusive movement of the animal will be absorbed by the numerical diffusion induced by the advection component. This is a drawback because it weakens the link between the discrete-space approximation and the continuous-space model.

Overall, there is not yet an efficient, robust way to include advection in spatial HMMs without suffering the drawbacks outlined above.

## S4.5 Conclusion

This brief appendix is intended to provide the necessary details for researchers to begin to understand the basic methods used to build and fit state-switching spatial HMMs with diffusive animal movement.

There are many extensions possible: restricting animal movement to certain cells (by setting elements of $\mathbf{Q}$ to zero), allowing for spatially-varying diffusion (by multipling rows of $\mathbf{G}$ by different values of `sd`), using HMM tools such as the Viterbi algorithm to make joint inference on location and behaviour, and incorporating measurement error (rather than assuming exact location is observed, up to grid cell resolution, as we have here).

## References


Descamps, S., Tarroux, A., Cherel, Y., Delord, K., Godø, O., Kato, A., Krafft, B., Lorentsen, S., Ropert-Coudert, Y., Skaret, G., and Varpe, Ø. (2016a). Data from: At-sea distribution and prey selection of Antarctic petrels and commercial krill fisheries.

Descamps, S., Tarroux, A., Cherel, Y., Delord, K., Godø, O. R., Kato, A., Krafft, B. A., Lorentsen, S.-H., Ropert-Coudert, Y., Skaret, G., et al. (2016b). At-sea distribution and prey selection of Antarctic petrels and commercial krill fisheries. *PloS one*, 11(8):e0156968.

Okubo, A. and Levin, S. A. (2001). *Diffusion and ecological problems: modern perspectives*. Springer.

Pedersen, M. W., Patterson, T. A., Thygesen, U. H., and Madsen, H. (2011a). Estimating animal behavior and residency from movement data. *Oikos*, 120(9):1281–1290.

Pedersen, M. W., Righton, D., Thygesen, U. H., Andersen, K. H., and Madsen, H. (2008). Geolocation of North Sea cod (Gadus morhua) using hidden markov models and behavioural switching. *Canadian Journal of Fisheries and Aquatic Sciences*, 65(11):2367–2377.

Pedersen, M. W., Thygesen, U. H., and Madsen, H. (2011b). Nonlinear tracking in a diffusion process with a bayesian filter and the finite element method. *Computational Statistics & Data Analysis*, 55(1):280–290.

Preisler, H. K., Ager, A. A., Johnson, B. K., and Kie, J. G. (2004). Modeling animal movements using stochastic differential equations. *Environmetrics*, 15(7):643–657.

Quarteroni, A. and Valli, A. (2008). *Numerical approximation of partial differential equations*. Springer Science & Business Media.

Sidje, R. B. (1998). Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software (TOMS)*, 24(1):130–156.

Thygesen, U. H., Pedersen, M. W., and Madsen, H. (2009). Geolocating fish using hidden Markov models and data storage tags. In *Tagging and Tracking of Marine Animals with Electronic Devices*, pages 277–293. Springer.

Zucchini, W., MacDonald, I. L., and Langrock, R. (2017). *Hidden Markov models for time series: an introduction using R, Second Edition*. CRC press.