Simulating and Fitting Animal movement using depmixS4

DepmixS4 is one of many package to fit hidden Markov models (HMM). In this example, we will demostrate how to simulate and fit covariate-dependent transition HMM using depmixS4.

```
tempdat <- data.frame(time=rep(1:24,50),y=NA)
head(tempdat)

##   time  y
## 1    1 NA
## 2    2 NA
## 3    3 NA
## 4    4 NA
## 5    5 NA
## 6    6 NA
```

Create a depmix object

```
library(depmixS4)

## Loading required package:  nnet
## Loading required package:  MASS
## Loading required package:  Rsolnp

tempmod <- (depmix(y~1  # response
  , nstates = 2 # number of states
  , transition=~cos(time/(2*pi))+sin(time/(2*pi)) # transition structure
  , family= gaussian()
  , data=tempdat
)
)

summary(tempmod)

## Initial state probabilties model
## pr1 pr2
## 0.5 0.5
##
## Transition model for state (component) 1
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
##                    St1 St2
## (Intercept)          0   0
## cos(time/(2 * pi))   0   0
```

```
## sin(time/(2 * pi))   0   0
## Probalities at zero values of the covariates.
## 0.5 0.5
##
## Transition model for state (component) 2
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
##                    St1 St2
## (Intercept)          0   0
## cos(time/(2 * pi))   0   0
## sin(time/(2 * pi))   0   0
## Probalities at zero values of the covariates.
## 0.5 0.5
##
##
## Response parameters
## Resp 1 : gaussian
##     Re1.(Intercept) Re1.sd
## St1               0      1
## St2               0      1
```

Set HMM parameters

```
setpars(tempmod, c(0.1, 0.9 # Initial prob parameters
  , 1 , 2 , 3 , 3 , 2 , 1 , 1 , 2 , 3 , 3 , 2 , 1 # Sinusoidal parameters for state switch
  , 0 , 1 , 2 , 1 # emission distribution parameters
  )
)
```

```
## Initial state probabilties model
## pr1 pr2
## 0.1 0.9
##
## Transition model for state (component) 1
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
##                    St1 St2
## (Intercept)          1   2
## cos(time/(2 * pi))   3   3
## sin(time/(2 * pi))   2   1
## Probalities at zero values of the covariates.
## 0.09893802 0.7310586
##
## Transition model for state (component) 2
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
```

```
##                     St1 St2
## (Intercept)          1   2
## cos(time/(2 * pi))   3   3
## sin(time/(2 * pi))   2   1
## Probalities at zero values of the covariates.
## 0.09893802 0.7310586
##
##
## Response parameters
## Resp 1 : gaussian
##     Re1.(Intercept) Re1.sd
## St1               0      1
## St2               2      1
```

```r
summary(tempmod)
```

```
## Initial state probabilties model
## pr1 pr2
## 0.5 0.5
##
## Transition model for state (component) 1
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
##                     St1 St2
## (Intercept)          0   0
## cos(time/(2 * pi))   0   0
## sin(time/(2 * pi))   0   0
## Probalities at zero values of the covariates.
## 0.5 0.5
##
## Transition model for state (component) 2
## Model of type multinomial (mlogit), formula: ~cos(time/(2 * pi)) + sin(time/(2 * pi))
## Coefficients:
##                     St1 St2
## (Intercept)          0   0
## cos(time/(2 * pi))   0   0
## sin(time/(2 * pi))   0   0
## Probalities at zero values of the covariates.
## 0.5 0.5
##
##
## Response parameters
## Resp 1 : gaussian
##     Re1.(Intercept) Re1.sd
## St1               0      1
## St2               0      1
```

Simulate hidden states and response data and put it back into a data frame. Strongly recommend simulating the response data using R, especially for user defined emission distrubtions.

```
source("simulate.R")
simdat <- simhmm(tempmod,nsim=1)
simdf <- data.frame(y=simdat@response[[1]][[1]]@y
  , states = simdat@states
  , time = tempdat$time)

head(simdf)

##            y states time
## 1  1.6318939      2    1
## 2 -0.9734612      2    2
## 3  0.7534086      1    3
## 4 -0.2798737      2    4
## 5 -0.2390554      1    5
## 6  0.4558390      2    6
```