R-Script Hidden Markov Models

This is run with an R kernel instead of a python kernel in Jupyter Notebook.

More reading:

- https://cran.r-project.org/web/packages/moveHMM/vignettes/moveHMM-guide.pdf
- https://stackoverflow.com/questions/57870575/install-and-run-r-kernel-for-jupyter-notebook

The goal for this notebook is to take the prepped GPS data from the 01-GPS PreProcessing notebook and then to use that to fit multiple different HMM that represent the deer movement model. Different numbers of behavioural states, and different landscape rasters are used to fit the HMM.

The different models are tested using Akaike's Information Criteriont (AIC) and Bayesian Information Criterion (BIC) to see which models are "Best".

The models tested are:

- Simple 2-state HMM: No landscape information included. Only 2 behaviour states modelled
- 2-state Landscape HMM: 2 behaviour states, raster info as covariate
- 3-state Landscape HMM: 2 behaviour states, raster info as covariate
- 2-state Home-Return HMM: 2 Behaviour states, no raster info, turn angles replaced with angle to centroid value

NOTE: This whole notebook takes hours to run. This isn't a very fast lib.

```
Loading required package: CircStats

Loading required package: MASS

Loading required package: boot


Attaching package: ‘arrow’


The following object is masked from ‘package:utils’:

    timestamp
```

ID	Deer ID	timestamp	sex	lat	lon	time_group	geometry	utm easting
<chr>	<dbl>	<dttm>	<chr>	<dbl>	<dbl>	<int>	<arrw_bnr>	<db
399253139925_gap_1	31	2017-02-02 00:00:12	f	39.22963	-76.88448	565	01, 01, 00, 00, 00, 0c, 38, f4, 1b, a0, c7, 38, 41, e7, da, 15, 9e, ef, f9, 3d, 41	337342
399253139925_gap_1	31	2017-02-02 01:00:28	f	39.23195	-76.88437	565	01, 01, 00, 00, 00, a4, ac, 86, 13, 75, c7, 38, 41, 36, 37, 8e, de, ef, fa, 3d, 41	337357

Take the GPS data and "prepare it", which calculates some variables and converts to data types required by the moveHMM package. Some different dataframes are created:

- **gps_data_simple**: This has deer ID's, step and turns, position info but not landscape raster info.
- **gps_data**: This has deer ID's, step and turns, position info and landscape raster info.
- **gps_data_home**: This has all the same fields as the "simple" dataframe, and also the distance and angles to the centroid

This takes a while to run... this isn't a multithreaded library.

The "prepData" command can take UTM or Lat/Lon position data, and the LLangle param can be TRUE, for great circle step and turn calculations, or FALSE for trigonometric calculations.

'ID' · 'step' · 'angle' · 'x' · 'y'

'ID' · 'step' · 'angle' · 'x' · 'y' · 'raster_value'

Step and Turn comparison

Let's double check that the step and turn calculated in the previous notebooks is the same, or pretty close, to the step and turns calculated by the MoveHMM library

A moveData: 6 × 5

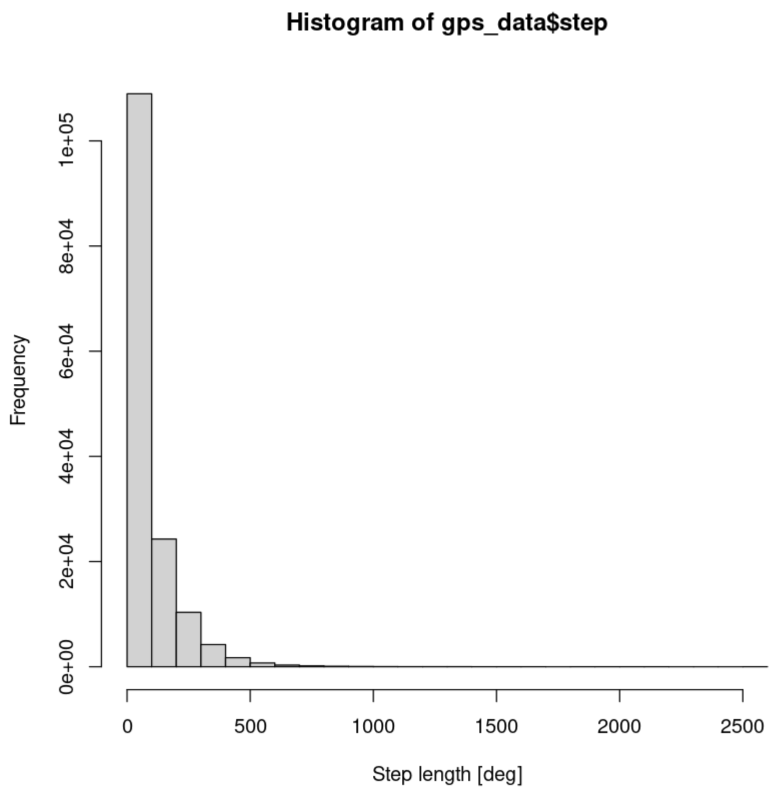
	ID	step	step_distance	angle	turn_angle
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	399253139925_gap_1	257.37552	259.84002	NA	-0.9321692
2	399253139925_gap_1	130.19103	129.57798	2.0515044	2.0603458
3	399253139925_gap_1	406.56052	408.93218	1.5184729	1.5027547
4	399253139925_gap_1	19.70712	19.60042	-2.5065172	-2.5066943
5	399253139925_gap_1	56.52751	56.32712	1.0567136	1.0731928
6	399253139925_gap_1	111.18051	110.40422	-0.2242944	-0.2267500

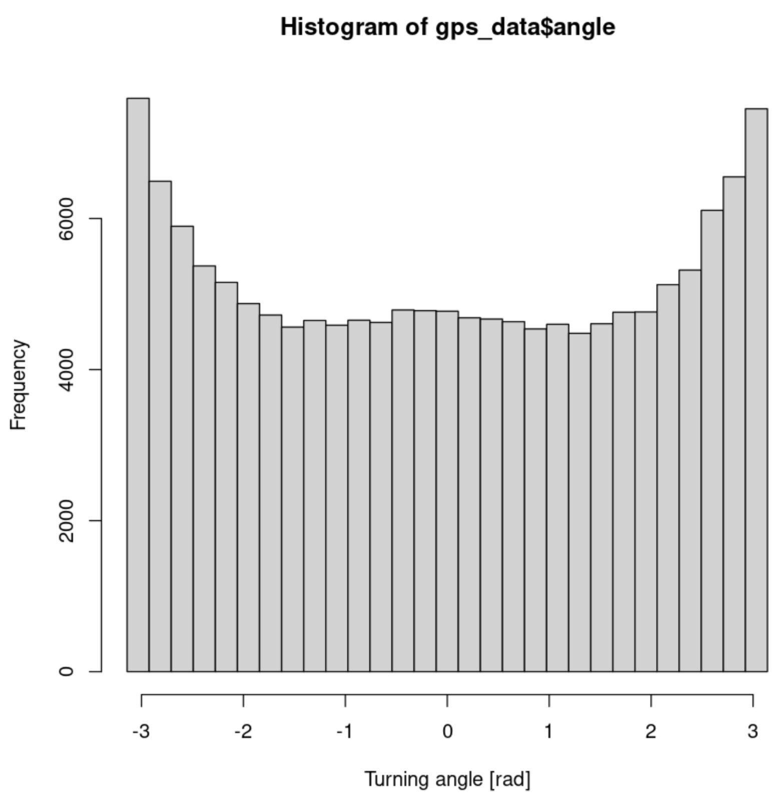
A moveData: 6 × 6

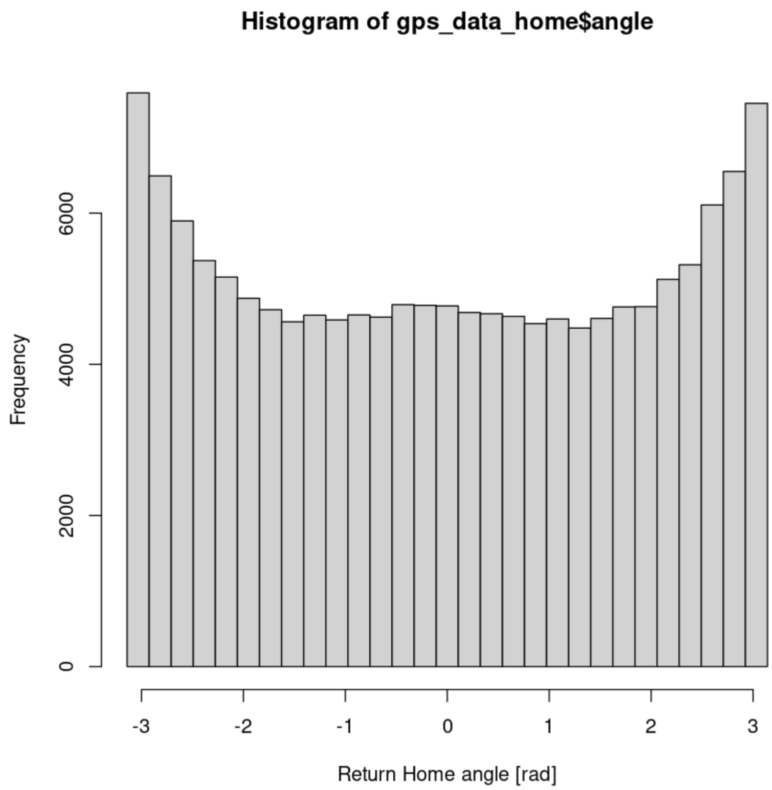
	ID	step	angle	x	y	raster_value
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	399253139925_gap_1	257.37552	NA	4343952	337342.8	95
2	399253139925_gap_1	130.19103	2.0515044	4344209	337357.2	53
3	399253139925_gap_1	406.56052	1.5184729	4344142	337469.1	87
4	399253139925_gap_1	19.70712	-2.5065172	4343783	337279.7	86
5	399253139925_gap_1	56.52751	1.0567136	4343791	337297.4	95
6	399253139925_gap_1	111.18051	-0.2242944	4343759	337343.9	65

A moveData: 6 × 7

	ID	x	y	step	angle	return_home_distance	raster_valu
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int
1	399253139925_gap_1	4343952	337342.8	257.37552	NA	230300.9	9
2	399253139925_gap_1	4344209	337357.2	130.19103	2.0515044	237949.5	5
3	399253139925_gap_1	4344142	337469.1	406.56052	1.5184729	236835.4	8
4	399253139925_gap_1	4343783	337279.7	19.70712	-2.5065172	224864.5	8
5	399253139925_gap_1	4343791	337297.4	56.52751	1.0567136	225248.0	9
6	399253139925_gap_1	4343759	337343.9	111.18051	-0.2242944	224655.5	6


Histogram of gps_data$step
Frequency
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
0
500
1000
1500
2000
2500
Step length [deg]


Histogram of gps_data$angle
Frequency
0
2000
4000
6000
-3
-2
-1
0
1
2
3
Turning angle [rad]



Some Different HMM Models:

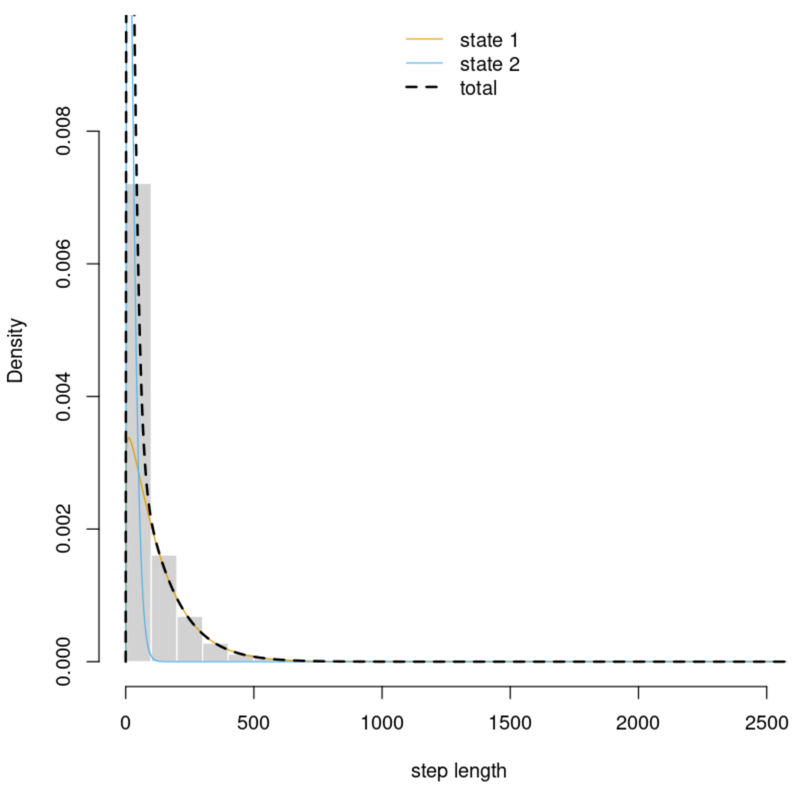
Simple 2 State Model

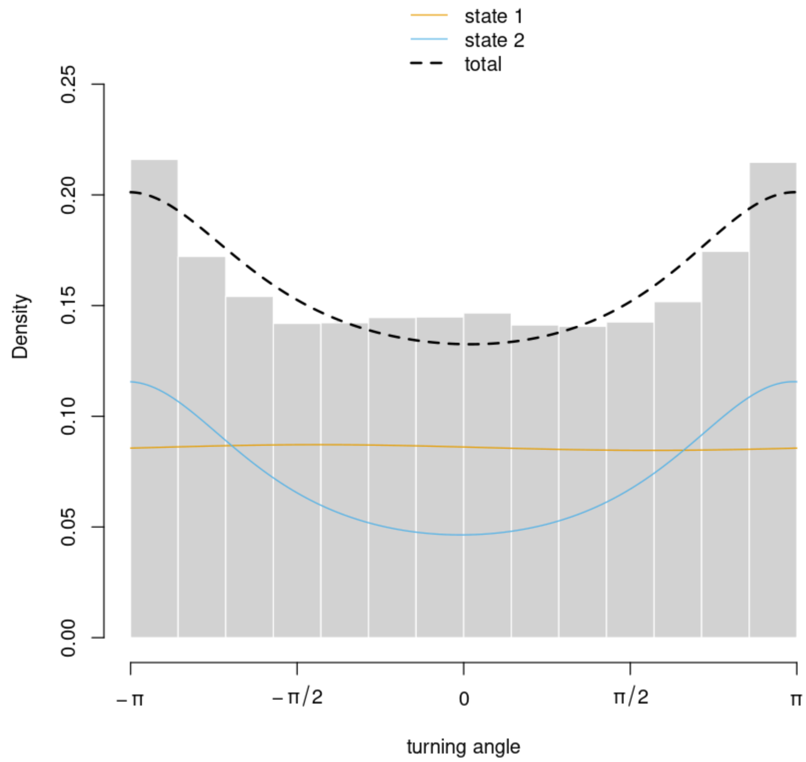
No covariates, 2 states.

When fitting a new model the initial parameters are VERY important. Values not close to the end results may prevent the model from converging or settling on a local minima.

```
Warning message in rbind(parts$upper, chars$ellip_v, parts$lower, deparse.level = 0
L):
“number of columns of result is not a multiple of vector length (arg 2)”
Warning message in rbind(parts$upper, chars$ellip_v, parts$lower, deparse.level = 0
L):
“number of columns of result is not a multiple of vector length (arg 2)”
Warning message in rbind(parts$upper, chars$ellip_v, parts$lower, deparse.level = 0
L):
“number of columns of result is not a multiple of vector length (arg 2)”
```

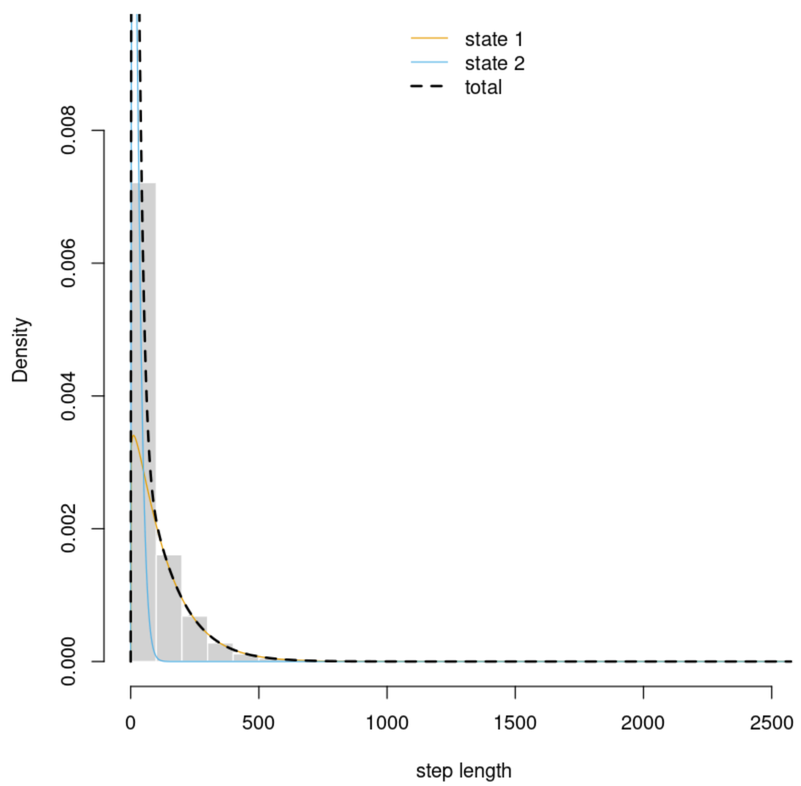
```
Value of the maximum log-likelihood: -1086359

Step length parameters:
----------------------
              state 1      state 2
shape     1.076009e+00 1.397961e+00
scale     1.328197e+02 2.721132e+01
zero-mass 1.482150e-05 7.525979e-05

Turning angle parameters:
------------------------
                  state 1   state 2
mean           -1.373675632 3.1108865
concentration  0.007442132 0.2242075

Regression coeffs for the transition probabilities:
--------------------------------------------------
             1 -> 2    2 -> 1
intercept -1.177283 -0.676602

Transition probability matrix:
-----------------------------
          [,1]      [,2]
[1,] 0.7644588 0.2355412
[2,] 0.3370201 0.6629799

Initial distribution:
--------------------
[1] 0.6218028 0.3781972

Decoding states sequence... DONE
```

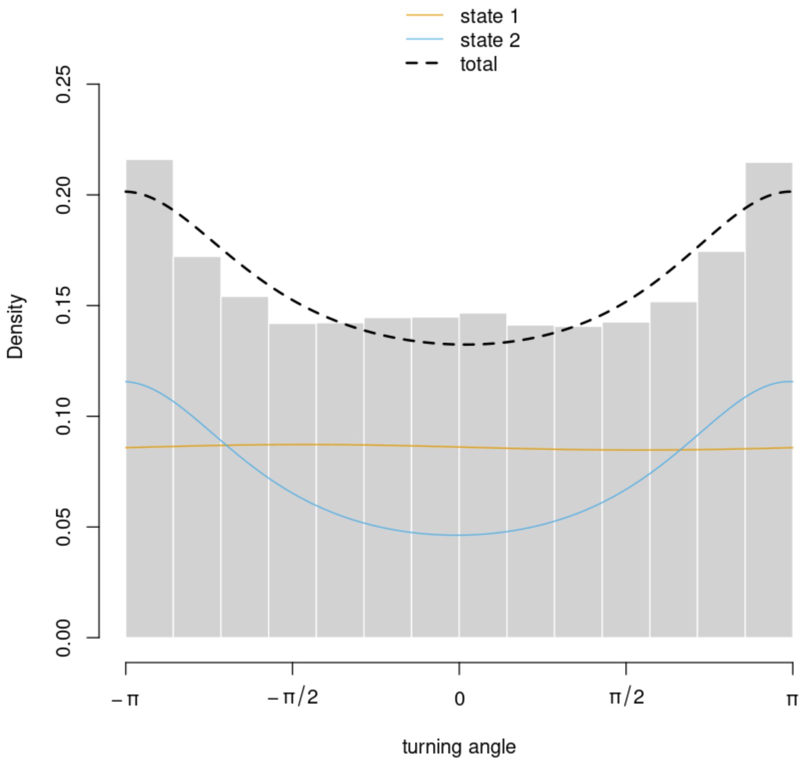




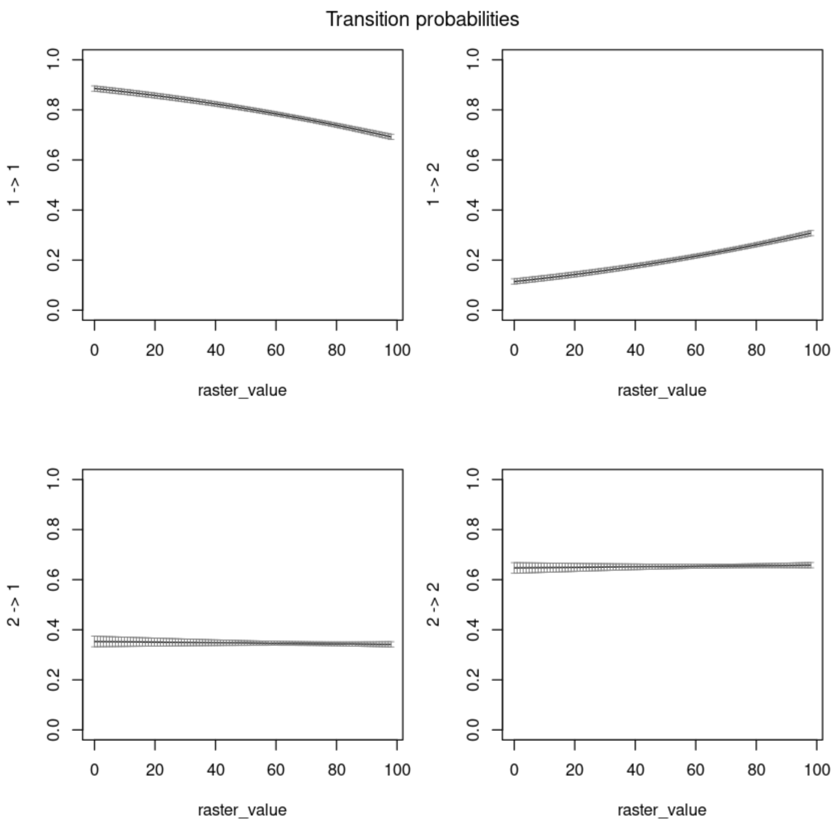
2 State Model, Landscape included

```
Decoding states sequence... DONE
```

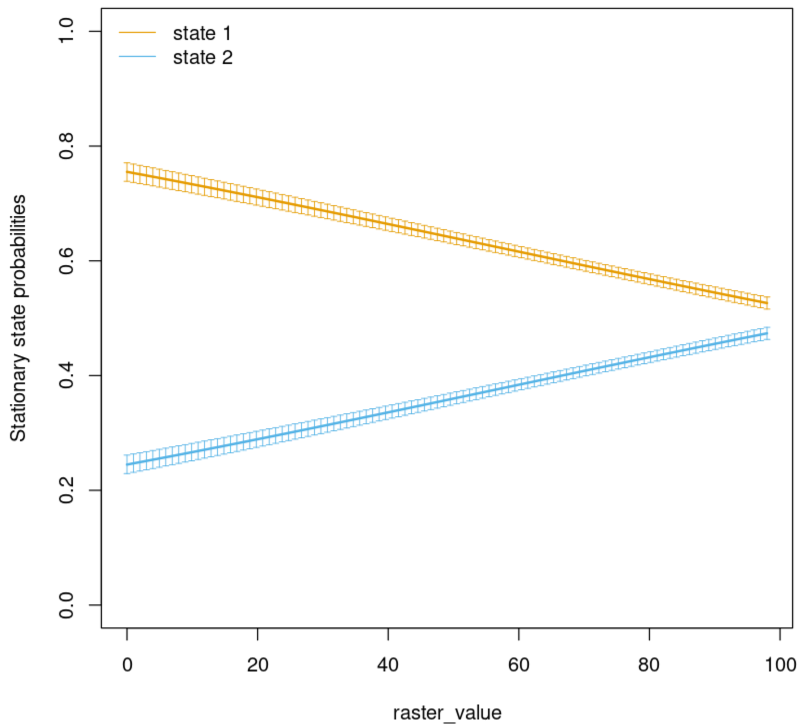




state 1
state 2
total
Density
0.00
0.05
0.10
0.15
0.20
0.25
−π
−π/2
0
π/2
π
turning angle



Transition probabilities
1 -> 1
1 -> 2
2 -> 1
2 -> 2
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
raster_value




```
Value of the maximum log-likelihood: -1086055

Step length parameters:
----------------------
               state 1      state 2
shape     1.073675e+00 1.411700e+00
scale     1.322886e+02 2.702609e+01
zero-mass 2.397647e-05 6.257293e-05

Turning angle parameters:
------------------------
                   state 1   state 2
mean          -1.454309789 3.1112617
concentration  0.007219257 0.2250314

Regression coeffs for the transition probabilities:
--------------------------------------------------
                  1 -> 2        2 -> 1
intercept    -2.04598566 -0.6061523294
raster_value  0.01259386 -0.0005008971

Initial distribution:
--------------------
[1] 0.6269333 0.3730667
```

HMM numbers to stick into Repast Model

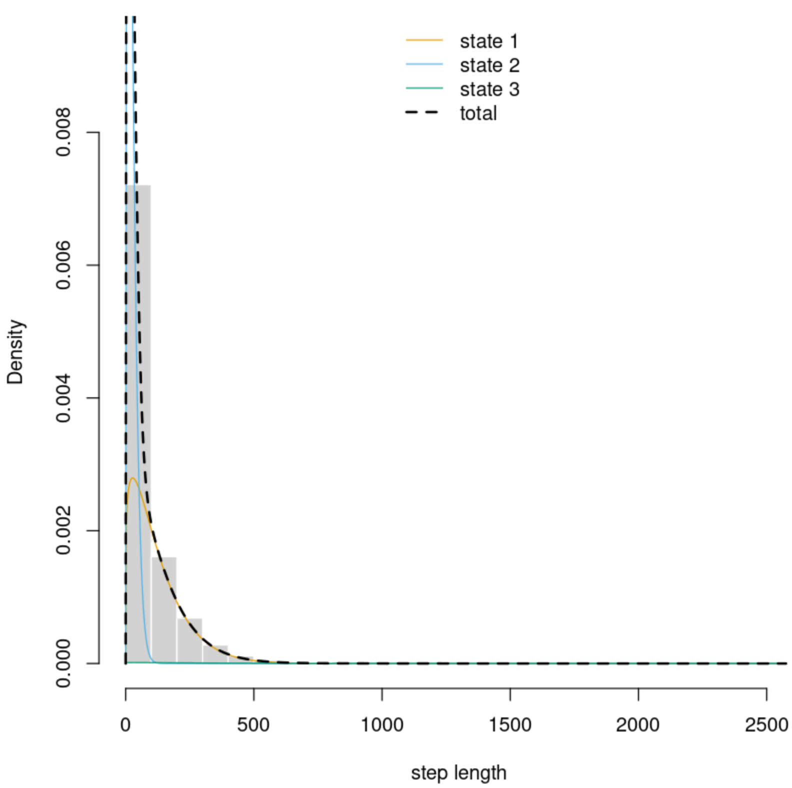
The HMM numbers that are needed for the Repast model. This gets stuck into the python model:

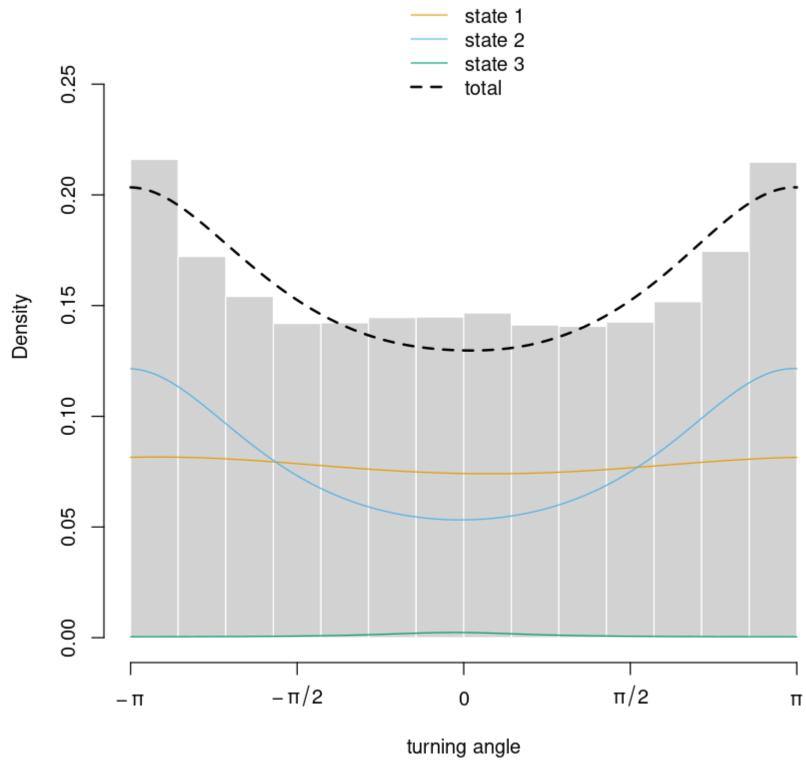
```python
class HMM_MoveModel_2_States(BaseMoveModel):
    '''
    Simple random movement model. Agents move around with a weibull/cauchy step and turn model.
    Not influenced by environmental, time or behaviour states. Just a pure random walk.
    '''
    def __init__(self, *args, **kwargs):
        self.movement_n_states = 2
        self.movement_params = [{'state': 0,
                                 'state_name': 'HMM State 0',
                                 'step_params':{'c': 1.38666,
                                                'loc': 1,
                                                'scale': 29.142},
                                 'turn_params':{'c': 0.23,
                                                'loc': -3.123,
                                                'scale': 1}},
                                {'state': 1,
                                 'state_name': 'HMM State 1',
                                 'step_params':{'c': 1.0378,
                                                'loc': 1,
                                                'scale': 148},
                                 'turn_params':{'c': 0.04,
                                                'loc': 0.1,
                                                'scale': 1}}
                                ] # Params for each different move state. List of N dictiona

        assert len(self.movement_params) == self.movement_n_states, "Too few movement params for number of m

        # HMM Covariate Intercept matrix
        self.hmm_covariate_intercept = np.asarray([-8.359713e-01, -2.08067466])
        # HMM Covariate Raster Coeff matrix
        self.hmm_covariate_coeff = np.asarray([5.790579e-05,  0.01360621])
```

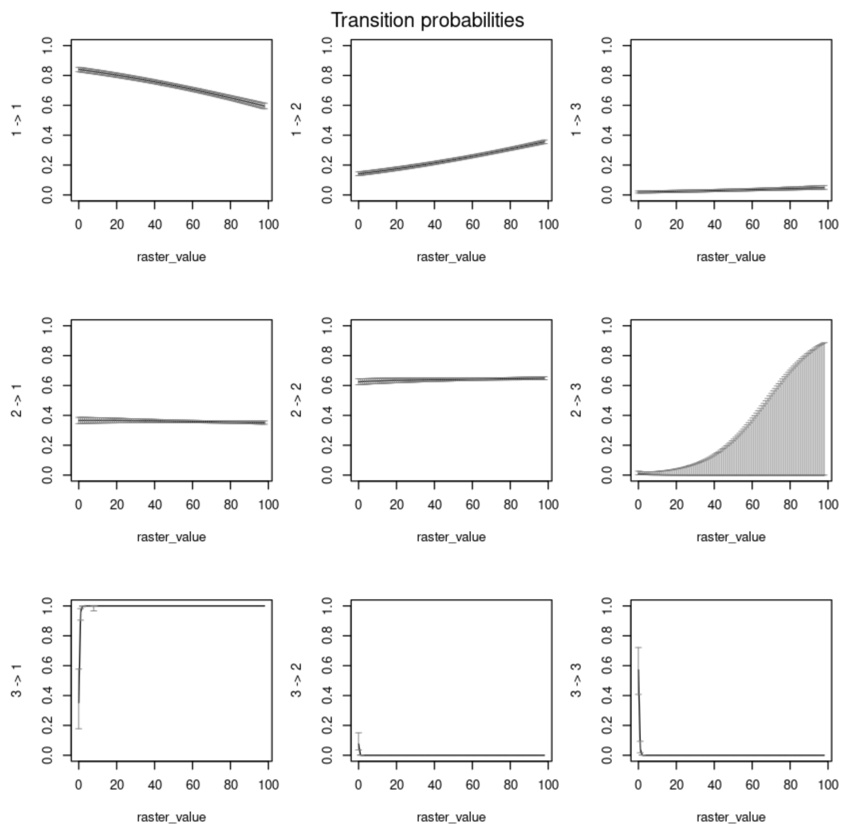
```
Value of the maximum log-likelihood: -1086055

Step length parameters:
----------------------
               state 1      state 2
shape     1.073675e+00 1.411700e+00
scale     1.322886e+02 2.702609e+01
zero-mass 2.397647e-05 6.257293e-05

Turning angle parameters:
------------------------
                   state 1   state 2
mean           -1.454309789 3.1112617
concentration   0.007219257 0.2250314

Regression coeffs for the transition probabilities:
--------------------------------------------------
                  1 -> 2          2 -> 1
intercept    -2.04598566 -0.6061523294
raster_value  0.01259386 -0.0005008971

Initial distribution:
--------------------
[1] 0.6269333 0.3730667
```

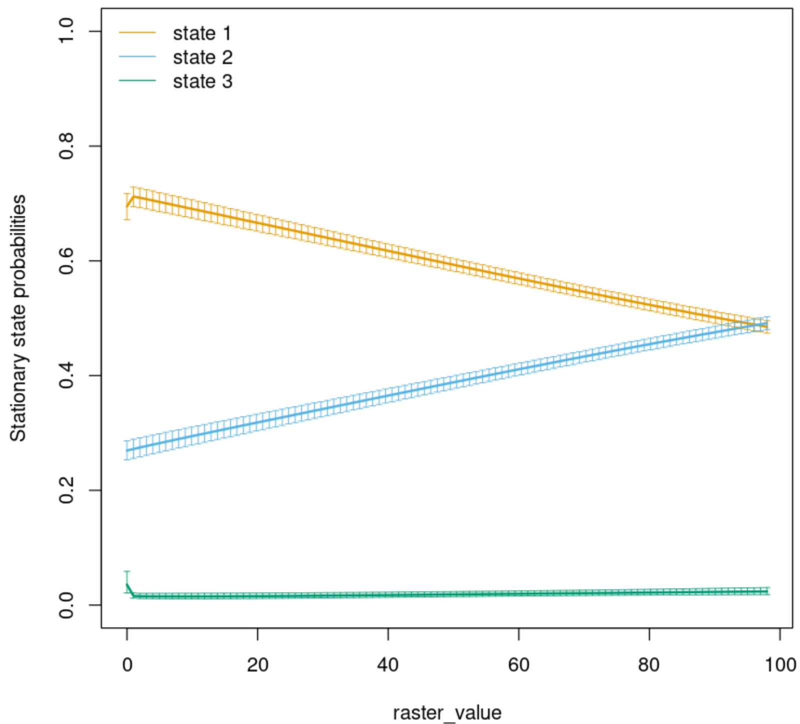
3 State Model, Landscape included

```
Value of the maximum log-likelihood: -1085192

Step length parameters:
----------------------
               state 1      state 2      state 3
shape     1.181699e+00 1.406792e+00 1.084333e+00
scale     1.327624e+02 2.645763e+01 3.448261e+02
zero-mass 1.143507e-05 4.952470e-05 4.961251e-04

Turning angle parameters:
------------------------
                  state 1   state 2     state 3
mean          -2.88861242 3.1114538 -0.08637389
concentration  0.02428817 0.2035553  0.41607627

Regression coeffs for the transition probabilities:
--------------------------------------------------
                 1 -> 2     1 -> 3        2 -> 1     2 -> 3     3 -> 1
intercept    -1.7812199 -3.8345379 -0.5354991864 -4.1782463 -0.4855525
raster_value  0.0129289  0.0136312 -0.0007961524 -0.1103394  3.6702500
                3 -> 2
intercept    -2.017667
raster_value -1.986867

Initial distribution:
--------------------
[1] 0.5820643 0.3968310 0.0211047

Decoding states sequence... DONE
```




state 1
state 2
state 3
total
Density
turning angle
0.00
0.05
0.10
0.15
0.20
0.25
$-\pi$
$-\pi/2$
0
$\pi/2$
π


Transition probabilities
1 -> 1
1 -> 2
1 -> 3
2 -> 1
2 -> 2
2 -> 3
3 -> 1
3 -> 2
3 -> 3
raster_value



$stepPar

A matrix: 3 × 3 of type dbl

	state 1	state 2	state 3
shape	1.181699e+00	1.406792e+00	1.084333e+00
scale	1.327624e+02	2.645763e+01	3.448261e+02
zero-mass	1.143507e-05	4.952470e-05	4.961251e-04

$anglePar

A matrix: 2 × 3 of type dbl

	state 1	state 2	state 3
mean	-2.88861242	3.1114538	-0.08637389
concentration	0.02428817	0.2035553	0.41607627

$beta

A matrix: 2 × 6 of type dbl

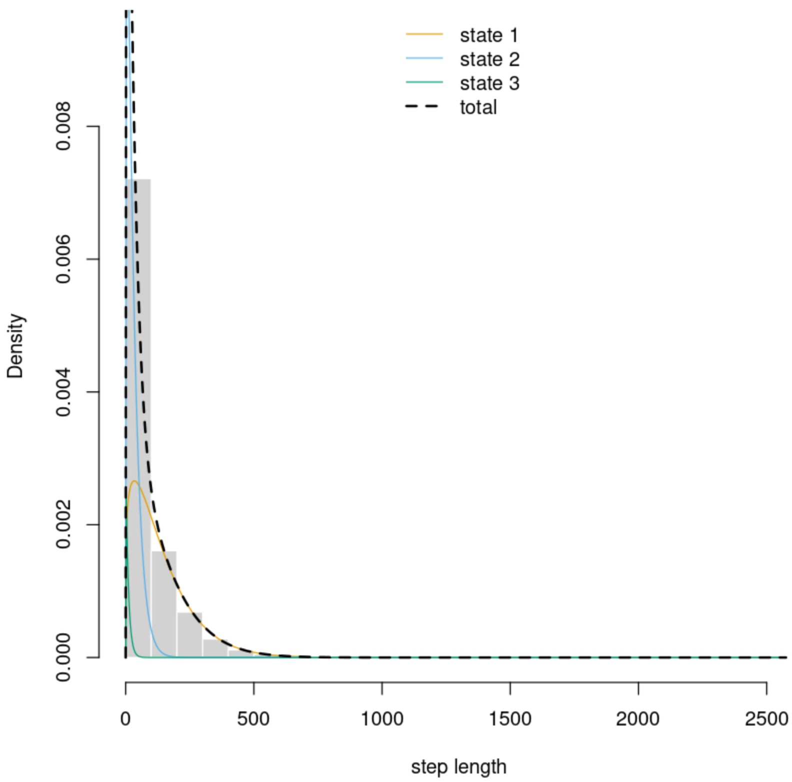
	1 -> 2	1 -> 3	2 -> 1	2 -> 3	3 -> 1	
intercept	-1.7812199	-3.8345379	-0.5354991864	-4.1782463	-0.4855525	-2
raster_value	0.0129289	0.0136312	-0.0007961524	-0.1103394	3.6702500	-1

$delta

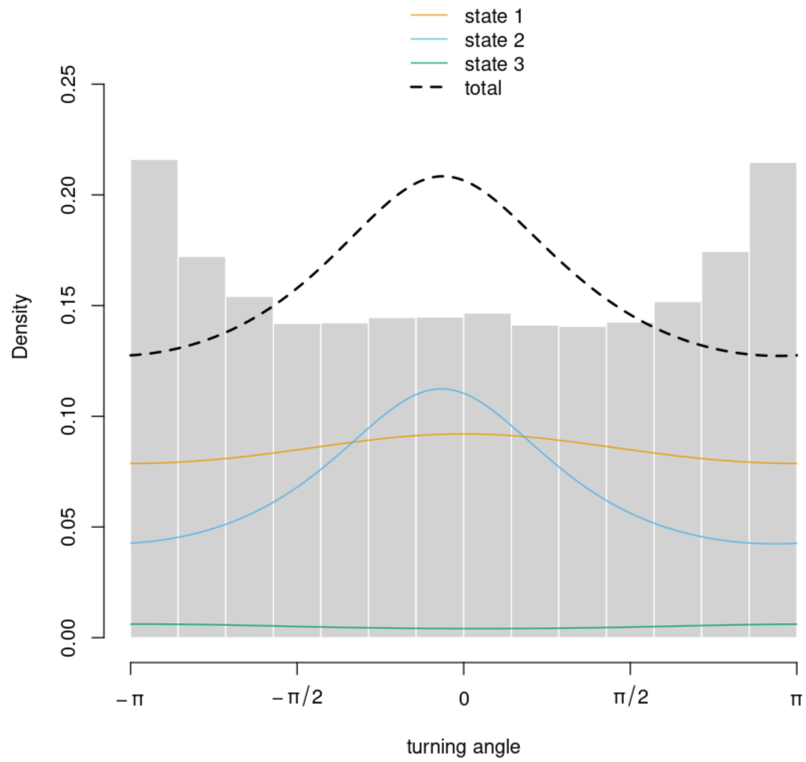
0.582064284268854 · 0.39683101371842 · 0.0211047020127256

3-States Return Home Model

```
Value of the maximum log-likelihood: -1099602

Step length parameters:
----------------------
              state 1      state 2      state 3
shape       1.19935645  1.100633281  1.000159229
scale     149.95998697 30.004283586 10.002050386
zero-mass   0.00999957  0.009999712  0.009999942

Turning angle parameters:
------------------------
                  state 1    state 2    state 3
mean          0.0008891589 -0.2103806 -3.0006785
concentration 0.0390528376  0.2390309  0.1001971

Regression coeffs for the transition probabilities:
--------------------------------------------------
                            1 -> 2        1 -> 3        2 -> 1        2 -> 3
intercept            -1.499974e+00 -1.5000064549 -1.499947e+00 -1.500011e+00
raster_value          3.267608e-03 -0.0003403201  3.000657e-03 -9.558267e-04
return_home_distance  1.353339e-06 -0.0005351144  6.856790e-06 -1.250872e-05
                            3 -> 1        3 -> 2
intercept            -1.4999912274 -1.4999979233
raster_value          0.0006502948  0.0001672923
return_home_distance  0.0024565235  0.0023512606

Initial distribution:
--------------------
[1] 0.3333344 0.3333345 0.3333311

Decoding states sequence... DONE
```




```
Error in svd(X): infinite or missing values in 'x'
Traceback:

1. plot.moveHMM(mod_3states_rh, ask = FALSE, plotTracks = FALSE, 
 .     plotCI = TRUE)
2. getPlotData(m, type = "tpm", format = "wide")
3. predictTPM(m = m, newData = tempCovs, returnCI = TRUE, alpha = 0.95)
4. ginv(m$mod$hessian)
5. svd(X)
6. stop("infinite or missing values in 'x'")
7. .handleSimpleError(function (cnd) 
 . {
 .     watcher$capture_plot_and_output()
 .     cnd <- sanitize_call(cnd)
 .     watcher$push(cnd)
 .     switch(on_error, continue = invokeRestart("eval_continue"), 
 .         stop = invokeRestart("eval_stop"), error = NULL)
 . }, "infinite or missing values in 'x'", base::quote(svd(X)))
```



But which is best?

Use the AIC values to decide which model is best.

Looks like the 3 state model performs the best, while the "return home" model is the worst.