`momentuHMM`: R package for analysis of telemetry data using generalized multivariate hidden Markov models of animal movement

Brett T. McClintock[1] and Théo Michelot[2]

---

[1]Marine Mammal Laboratory
Alaska Fisheries Science Center
NOAA National Marine Fisheries Service
Seattle, U.S.A.
Email: brett.mcclintock@noaa.gov

[2]School of Mathematics and Statistics
University of Sheffield
Sheffield, U.K.

November 21, 2018

Summary

1. Discrete-time hidden Markov models (HMMs) have become an immensely popular tool for inferring latent animal behaviors from telemetry data, largely because they are relatively fast and easy to implement when data streams are observed without error and at regular time intervals. While movement HMMs typically rely solely on location data, auxiliary biotelemetry and environmental data are powerful and readily-available resources for incorporating much more behavioral realism and inferring ecological relationships that would otherwise be difficult or impossible to infer from location data alone. However, there is a paucity of generalized user-friendly software available for implementing (multivariate) HMMs of animal movement. Furthermore, location measurement error, temporal irregularity, and other forms of missing data are often pervasive in telemetry studies (particularly in marine systems).

2. Here we provide a guide to using an open-source R package, `momentuHMM` version 1.4.3, that addresses many of the deficiencies in existing software. Features for multivariate HMMs in `momentuHMM` (pronounced "momentum") include: 1) tools for data pre-processing and visualization; 2) user-specified probability distributions for an unlimited number of data streams and latent behavior states, such as those based on location (e.g., step length, turning angle) and auxiliary biotelemetry data (e.g., from pressure, conductivity, heart rate, or motion sensors); 3) biased and correlated random walk movement models, including "activity centers" associated with attractive or repulsive forces; 4) user-specified design matrices and constraints for covariate modelling of initial distribution, state transition probability, and probability distribution parameters using linear model formulas familiar to most R users; 5) multiple imputation methods that account for observation error attributable to measurement error and temporally-irregular or missing data; 6) seamless integration of spatio-temporal covariate raster data; 7) cosinor and spline regression formulas for cyclical (e.g., daily, seasonal) and other complicated patterns; 8) model checking and selection; and 9) data simulation capabilities for study design, power analyses and assessing model performance, including simulation of location data subject to temporal irregularity and/or measurement error.

3. After providing a brief introduction to (multivariate) HMMs for telemetry data, we demonstrate some of the capabilities of `momentuHMM` using real-world examples. This brief tutorial includes workflows for data formatting, model specification, model fitting,

and diagnostics.

4. While many of the features of `momentuHMM` were motivated by animal movement data, the package can be used for analyzing any type of data that is amenable to (multivariate) HMMs. Practitioners interested in additional features for `momentuHMM` are encouraged to contact the authors.

Key-words animal biotelemetry, biologging, `crawl`, `moveHMM`, state-space model, state-switching

1 Introduction

Discrete-time hidden Markov models (HMMs) have become immensely popular for the analysis of animal telemetry data (e.g. Morales *et al.* 2004; Jonsen *et al.* 2005; Langrock *et al.* 2012; McClintock *et al.* 2012). In short, an HMM is a time series model composed of a (possibly multivariate) observation process $(\mathbf{Z}_1, \ldots, \mathbf{Z}_T)$, in which each data stream is generated by N state-dependent probability distributions, and where the unobservable (hidden) state sequence $(S_t \in \{1, \ldots, N\}, t = 1, \ldots, T)$ is assumed to be a Markov chain. The state sequence of the Markov chain is governed by (typically first-order) state transition probabilities, $\gamma_{ij}^{(t)} = \Pr(S_{t+1} = j \mid S_t = i)$ for $i, j = 1, \ldots, N$, and an initial distribution $\boldsymbol{\delta}^{(0)}$. The likelihood of an HMM can be succinctly expressed using the forward algorithm:

$$\mathcal{L} = \boldsymbol{\delta}^{(0)} \boldsymbol{\Gamma}^{(1)} \mathbf{P}(\mathbf{z}_1) \boldsymbol{\Gamma}^{(2)} \mathbf{P}(\mathbf{z}_2) \boldsymbol{\Gamma}^{(3)} \cdots \boldsymbol{\Gamma}^{(T-1)} \mathbf{P}(\mathbf{z}_{T-1}) \boldsymbol{\Gamma}^{(T)} \mathbf{P}(\mathbf{z}_T) \mathbf{1}^N, \tag{1}$$

where $\boldsymbol{\Gamma}^{(t)} = \left(\gamma_{ij}^{(t)}\right)$ is the $N \times N$ transition probability matrix, $\mathbf{P}(\mathbf{z}_t) = \text{diag}(p_1(\mathbf{z}_t), \ldots, p_N(\mathbf{z}_t))$, $p_s(\mathbf{z}_t)$ is the conditional probability density of $\mathbf{Z}_t$ given $S_t = s$, and $\mathbf{1}^N$ is a N-vector of ones (for a thorough introduction to HMMs see Zucchini *et al.* 2016).

One of the most common discrete-time animal movement HMMs for telemetry location data is composed of two data streams, step length and turning angle (or bearing), which are calculated for each of the T time steps from the temporally-regular observations of an animal's position, (x_t, y_t), for $t = 1, \ldots, T + 1$ (e.g. Morales *et al.* 2004; Langrock *et al.* 2012; McClintock *et al.* 2012). Step length (l_t) is typically calculated as the Euclidean distance between the locations (x_t, y_t) and (x_{t+1}, y_{t+1}), while turning angle (ϕ_t) is calculated as the change in bearing $(b_t = \text{atan2}(y_{t+1} - y_t, x_{t+1} - x_t))$ between

the intervals $[t-1, t]$ and $[t, t+1]$ (e.g. $\phi_t = 0$ if $b_{t-1} = b_t$). For this HMM composed of 2 data streams, $\mathbf{z}_t = (l_t, \phi_t)$, and, conditional on the latent state S_t, independent probability distributions are typically assumed for each stream; that is, $p_s(\mathbf{z}_t) = p_s(l_t)p_s(\phi_t)$. Some common probability distributions for the step length data stream are the gamma or Weibull distributions, while the wrapped Cauchy or von Mises distributions are often employed for turning angle or bearing. For a fitted HMM, the Viterbi algorithm is used to compute the most likely sequence of underlying states (Zucchini *et al.* 2016). In movement HMMs, the states are often considered as proxies for animal behaviour.

While HMMs for animal movement based solely on location data are somewhat limited in the number and type of biologically-meaningful movement behavior states they are able to accurately identify, advances in biologging technology are now allowing the collection of valuable auxiliary biotelemetry data (e.g., dive activity, accelerometer, heart rate, stomach temperature), which, when combined with location data, allow for multivariate HMMs that can incorporate much more behavioral realism and facilitate inferences about complex ecological relationships that would otherwise be difficult or impossible to infer from location data alone (e.g. McClintock *et al.* 2013; DeRuiter *et al.* 2017; McClintock *et al.* 2017). Multivariate HMMs that utilize both location and auxiliary biotelemetry data can facilitate the identification of additional states that go beyond the $N = 2$ state approaches that are most frequently used by practitioners. For example, the most widely used 2-state HMMs for animal movement include "encamped" (or "foraging") and "exploratory" (or "transit") states characterized by area-restricted-search-type movements (shorter step lengths with little to no directional persistence) and migratory-type movements (longer step lengths with high directional persistence), respectively (Morales *et al.* 2004; Jonsen *et al.* 2005). However, very different behaviors can exhibit similar horizontal trajectories. For example, for herbivores such as North American elk (Morales *et al.* 2004) or central-place foragers such as harbour seals (McClintock *et al.* 2013), the horizontal trajectories of "resting" and "foraging" movements can be very difficult to distinguish. Standard 2-state HMMs based solely on horizontal trajectory will tend to lump these behaviors together, and this could have unintended consequences if, for example, one intends to use the estimated state sequences to identify foraging habitat. In order to tweeze apart distinct behaviors with similar horizontal trajectories, additional states can be informed by auxiliary information (such as mandible accelerometer or dive data), incorporated as additional data

stream(s) in a multivariate HMM.

When data streams are observed without error and at regular time intervals, a major advantage of HMMs is the relatively fast and efficient maximization of the likelihood using the forward algorithm (Eq. 1). However, location measurement error is rarely non-existent in animal-borne telemetry studies and depends on both the device and the system under study. For example, GPS errors are typically less than 50m, but Argos errors can exceed 10km (e.g. Costa *et al.* 2010). An extreme case of missing data can arise when location data are obtained with little or no temporal regularity, as in many marine mammal telemetry studies (e.g. Jonsen *et al.* 2005), such that few (if any) observations align with the regular time steps required by discrete-time HMMs. When explicitly accounting for uncertainty attributable to location measurement error, temporally-irregular observations, or other forms of missing data, one must typically fit (multivariate) HMMs using computationally-intensive (and often time-consuming) model fitting techniques such as Markov chain Monte Carlo (Jonsen *et al.* 2005; McClintock *et al.* 2012). However, complex analyses requiring novel statistical methods and custom model-fitting algorithms are not practical for many practitioners.

While statisticians have been applying HMMs to telemetry data for decades, R (R Core Team 2017) packages such as `bsam` (Jonsen *et al.* 2005), `moveHMM` (Michelot *et al.* 2016), and `swim` (Whoriskey *et al.* 2017) have recently helped make these models of animal movement behavior more accessible to the practitioners that are actually conducting telemetry studies. These advances represent important steps toward making HMMs of animal movement more accessible, but the models that can currently be implemented using existing software remain limited in many key respects. For example, existing HMM software for animal movement is limited to two data streams based solely on location data (e.g. step length and turning angle), and while `moveHMM` allows for a user-specified number of latent behavioral states (`bsam` and `swim` are limited to $N = 2$ states), it is typically difficult to identify >2 biologically-meaningful behavior states from only 2 data streams (e.g. Morales *et al.* 2004; Beyer *et al.* 2013; McClintock *et al.* 2014). Both `moveHMM` and `swim` are designed for temporally-regular (or linearly-interpolated) location data with negligible measurement error, but the realities of animal-borne telemetry often yield temporally-irregular location data subject to error (particularly in aquatic environments). Other notable deficiencies of existing software include limited abilities to incorporate spatio-temporal environmental or

individual covariates on parameters, biased (or directed) movements in response to attractive or repulsive forces (e.g. McClintock *et al.* 2012; Langrock *et al.* 2014), cyclical (e.g. daily, seasonal) and other more complicated behavioral patterns, or constraints on parameters.

To address these deficiencies in existing software, we developed a user-friendly R package, `momentuHMM` (Maximum likelihood analysis Of animal MovemENT behavior Using multivariate Hidden Markov Models), intended for practitioners wishing to implement more flexible and realistic (multivariate) HMM analyses of animal movement while accounting for common challenges associated with telemetry data (McClintock & Michelot 2018). Features for multivariate HMM analyses in `momentuHMM` include: 1) tools for data pre-processing and visualization; 2) user-specified probability distributions for an unlimited number of data streams and latent behavior states; 3) biased and correlated random walk movement models, including "activity centers" associated with attractive or repulsive forces (e.g. McClintock *et al.* 2012); 4) user-specified design matrices and constraints for covariate modelling of state transition probability and probability distribution parameters using linear model formulas familiar to most R users; 5) multiple imputation methods that account for observation error attributable to measurement error and temporally-irregular or missing data (Hooten *et al.* 2017; McClintock 2017); 6) seamless integration of spatio-temporal environmental covariate data (e.g., wind direction, forest cover, sea ice concentration) using the `raster` package (Hijmans 2016b); 7) cosinor (e.g. Cornelissen 2014) and spline regression formulas for cyclical and other complicated behavioral patterns; 8) model checking and selection; and 9) data simulation capabilities for study design, power analyses and assessing model performance, including simulation of location data subject to temporal irregularity and/or measurement error.

In the following tutorial, we demonstrate some of the capabilities of `momentuHMM` using real-world examples, including an example of periodic cycles in African elephant movement, a 3-state ("resting", "foraging", "transit") northern fur seal example incorporating auxiliary dive activity data (McClintock *et al.* 2014), a loggerhead turtle example relating "foraging" and "transit" movements to ocean surface currents, a 5-state grey seal example incorporating biased movements toward haul-out and foraging locations (McClintock *et al.* 2012), a 4-state ("outbound", "searching", "foraging", "inbound") southern elephant seal example with biased movements toward and away from

a colony (Michelot *et al.* 2017), a 3-state ("resting", "foraging", "transit") harbour seal example using population-level constraints on movement parameters (McClintock *et al.* 2013), and a 6-state northern fulmar example incorporating biased movements relative to both static (i.e. colony) and dynamic (i.e. fishing vessels) activity centers (Pirotta *et al.* 2018). Using simulated data, we also demonstrate how the group dynamic model of Langrock *et al.* (2014) can be implemented using `momentuHMM`. This brief tutorial includes workflows for data formatting, model specification, model fitting, and diagnostics. While many of the features of `momentuHMM` were motivated by animal movement data, the package can be used for analyzing any type of data that is amenable to (multivariate) HMMs. Additional information, including help files, data, examples, and package usage is available by downloading the `momentuHMM` package from CRAN (`cran.r-project.org`) or Github (`github.com/bmcclintock/momentuHMM`). This article describes `momentuHMM` version 1.4.3.

2 momentuHMM overview

Before delving into some of the finer details, we will first provide an overview of the main features and functions of `momentuHMM` (pronounced "momentum"). While space is limited in this tutorial, further details on implementation can be found in the package's documentation and vignette. The workhorse functions of `momentuHMM` are listed in Table 1. Usage of several of these functions (e.g. `fitHMM`, `prepData`, `simData`) is deliberately very similar to equivalent functions in `moveHMM` (Michelot *et al.* 2016) , but the `momentuHMM` arguments for these functions have been generalized and expanded to accommodate a more flexible framework for data pre-processing, model specification, parameterization, and simulation. R users already familiar with `moveHMM` will therefore likely find it easy to immediately begin using `momentuHMM`.

One of the key features of `momentuHMM` is the ability to include an unlimited number of HMM data streams (e.g. step length, turning angle, dive activity, heart rate) arising from a broad range of commonly used probability distributions (e.g. beta, gamma, normal, Poisson, von Mises, Weibull). Any of the parameters of the probability distributions used for the observed data can be modelled as a function of environmental and individual covariates using link functions (Table 2). For any given "natural scale" (or "real scale") probability distribution parameter θ, all of the link functions (g) in

Table 1. *Workhorse functions for the R package momentuHMM.*

Function	Description
`crawlMerge`	Merge `crawlWrap` output with additional data streams or covariates
`crawlWrap`	Fit `crawl` models and predict temporally-regular locations
`fitHMM`	Fit a (multivariate) HMM to the data
`MIfitHMM`	Fit (multivariate) HMMs to multiple imputation data
`MIpool`	Pool `momentuHMM` model results across multiple imputations
`plot.crwData`	Plot `crawlWrap` output
`plot.miSum`	Plot summaries of multiple imputation `momentuHMM` models
`plot.momentuHMM`	Plot summaries of `momentuHMM` models
`plot.momentuHMMData`	Plot summaries of selected data streams and covariates
`plotPR`	Plot time series, qq-plots and sample ACFs of pseudo-residuals
`plotSat`	Plot locations on satellite image
`plotSpatialCov`	Plot locations on raster image
`plotStates`	Plot the (Viterbi-)decoded states and state probabilities
`plotStationary`	Plot stationary state probabilities
`prepData`	Pre-process data streams and covariates
`pseudoRes`	Calculate pseudo-residuals for `momentuHMM` models
`simData`	Simulate data from a (multivariate) HMM
`stateProbs`	State probabilities for each time step
`viterbi`	Most likely state sequence (using the Viterbi algorithm)

momentuHMM are of the general form $g(\boldsymbol{\theta}) = \mathbf{X}_\theta \boldsymbol{\beta}_\theta$, where $\mathbf{X}_\theta$ is the $T \times K$ design matrix (composed of K covariates) and $\boldsymbol{\beta}_\theta$ is the correponding K-vector of "working scale" (or "beta scale") parameters for θ. For example, suppose step length is assumed to have a gamma distribution, $l_t \mid S_t = s \sim \text{gamma}(\mu_s, \sigma_s)$. In momentuHMM, the natural scale parameters for the gamma distribution are the (state-dependent) step length mean ($\mu_s > 0$) and standard deviation ($\sigma_s > 0$). Because both of these parameters must be positive, the log link function is a natural choice for modelling these parameters as a function of covariates, e.g., $\log(\boldsymbol{\mu}) = \mathbf{X}_\mu \boldsymbol{\beta}_\mu$ and $\log(\boldsymbol{\sigma}) = \mathbf{X}_\sigma \boldsymbol{\beta}_\sigma$.

The state transition probabilities ($\boldsymbol{\Gamma}^{(t)}$) and initial distribution ($\boldsymbol{\delta}^{(0)}$) can also be modelled as functions of covariates, using a multinomial logit link, as described e.g. by Michelot *et al.* (2016). Permissable R classes for covariates include `numeric`, `integer`, or `factor`. Factors can be particularly useful for specifying models with individual- or group-level (e.g. sex or age class) effects on state transition and probability distribution parameters. Spatio-temporal covariates can also be of classes `rasterLayer`, `rasterStack`, or `rasterBrick` (Hijmans 2016b), in which case momentuHMM automatically extracts the appropriate covariate values from the raster based on the time and location of each observation (see example in section 3.3).

2.1 Data preparation and visualization

For temporally-regular location data with negligible measurement error, the `prepData` function is used to create a `momentuHMMData` object that can be used for data visualization and further analysis. The arguments for `prepData` include:

- `data` A data frame with $T + 1$ rows including optionally a field 'ID' (identifiers for different individuals), coordinates from which step length ('step') and turning angle ('angle') data streams are to be calculated, any additional data streams, and any covariates identified in the `covNames` and `angleCovs` arguments. Altervatively, `data` can be a `crwData` object returned by `crawlWrap`.

- `type` Coordinate type; 'UTM' if easting-northing or 'LL' if longitude-latitude.

- `coordNames` Names of the two coordinate columns in `data`. If `coordNames=NULL` then step lengths, turning angles, and any location-based covariates (i.e., those specified by `spatialCovs`, `centers`, `centroids`, and `angleCovs`) are not calculated.

Table 2. *Data stream (z) probability distributions, natural parameters, and default link functions for covariate modelling. Probability distributions with positive support can be zero-inflated (with additional zero-mass parameters), while the beta distribution can be zero- and/or one-inflated (with additional one-mass parameters). If user-specified bounds are provided, then custom link functions are used instead of the defaults (see package documentation for further details). If both zero- and one-inflation are included, then a multinomial logit link is used because these probabilities must sum to less than one (in this case, any user-specified bounds for the zero- and one-inflation parameters are ignored). If circular-circular regression is specified for the mean of angular distributions ("vm" and "wrpcauchy"), then a link function based on Rivest* et al. *(2016) is used. The von Mises consensus distribution ("vmConsensus") is a von Mises circular-circular regression model where the concentration parameter depends on the level of agreement among short-term directional persistence and angular covariates. Users seeking additional probability distributions are encouraged to contact the authors.*

Distribution	Support	Parameters	Link function[1]
Bernoulli ("`bern`")	$z_t \in \{0, 1\}$	`prob` $\in (0, 1)$	logit
Beta ("`beta`")	$z_t \in (0, 1)$	`shape1` > 0	log
		`shape2` > 0	log
		`zero-mass` $\in (0, 1)$	logit
		`one-mass` $\in (0, 1)$	logit
Exponential ("`exp`")	$z_t > 0$	`rate` > 0	log
		`zero-mass` $\in (0, 1)$	logit
Gamma ("`gamma`")	$z_t > 0$	`mean` > 0	log
		`sd` > 0	log
		`zero-mass` $\in (0, 1)$	logit
Log normal ("`lnorm`")	$z_t > 0$	`location` $\in \mathbb{R}$	identity
		`scale` > 0	log
		`zero-mass` $\in (0, 1)$	logit
Normal ("`norm`")	$z_t \in \mathbb{R}$	`mean` $\in \mathbb{R}$	identity
		`sd` > 0	log
Poisson ("`pois`")	$z_t \in \{0, 1, \ldots\}$	`lambda` > 0	log
Von Mises ("`vm`")	$z_t \in (-\pi, \pi]$	`mean` $\in (-\pi, \pi]$	tan(`mean`/2)
		`concentration` > 0	log
Von Mises ("`vmConsensus`")	$z_t \in (-\pi, \pi]$	`mean` $\in (-\pi, \pi]$	Rivest *et al.* (2016)
		`kappa` > 0	log
Weibull ("`weibull`")	$z_t > 0$	`shape` > 0	log
		`scale` > 0	log
		`zero-mass` $\in (0, 1)$	logit
Wrapped Cauchy ("`wrpcauchy`")	$z_t \in (-\pi, \pi]$	`mean` $\in (-\pi, \pi]$	tan(`mean`/2)
		`concentration` $\in (0, 1)$	logit

[1]Link functions (g) relate natural scale parameters ($\boldsymbol{\theta}$) to a $T \times K$ design matrix ($\mathbf{X}$) and $K-$vector of working scale parameters ($\boldsymbol{\beta} \in \mathbb{R}^K$) such that $g(\boldsymbol{\theta}) = \mathbf{X}\boldsymbol{\beta}$.

- `covNames` Character vector indicating the names of any covariates in `data`. Any variables in `data` (other than "ID") that are not identified in `covNames` or `angleCovs` are assumed to be data streams.

- `spatialCovs` List of Raster-class objects (Hijmans 2016b) containing spatio-temporally referenced covariates. Covariates specified by `spatialCovs` are extracted from the raster layer(s) based on the location data. Raster stacks may also be included, in which case the appropriate z values (e.g. time, date) must also be included in `data`.

- `centers` 2-column matrix providing the coordinates for any activity centers (e.g., potential centers of attraction or repulsion) from which distance and angle covariates will be calculated based on the location data and returned in the `momentuHMMData` object.

- `centroids` List where each element is a data frame containing the x-coordinates ('x'), y-coordinates ('y'), and times for a centroid (i.e., a dynamic activity center for which the coordinates can change over time) from which distance and angle covariates will be calculated based on the location data and returned in the `momentuHMMData` object.

- `angleCovs` Character vector indicating the names of any circular-circular regression angular covariates in `data` or `spatialCovs` that need conversion from standard direction (in radians relative to the x-axis) to turning angle (relative to previous movement direction).

Summary plots of the `momentuHMMData` object returned by `prepData` can be created for any data stream or covariate using the generic `plot` function.

If location data are temporally-irregular or subject to measurement error, then they are not suitable for `prepData`. In this case, `momentuHMM` can be used to perform a 2-stage multiple imputation approach (McClintock 2017). We discuss this pragmatic approach to incorporating uncertainty attributable to observation error and temporal irreglarity into multivariate HMM analyses in section 2.4.

2.2 HMM specification and fitting

Once a `momentuHMMData` object has been created using `prepData`, then the data are ready to be passed to the generalized multivariate HMM-fitting function `fitHMM`. There are many different options for specifying HMMs using `fitHMM`, so here we will only focus on several of the most important and useful features (further details of all `fitHMM` arguments are in the package documentation). The bare essentials of `fitHMM` include the arguments:

- `data` A `momentuHMMData` object
- `nbStates` Number of latent states (N)
- `dist` A named list indicating the probability distributions of the data streams.
- `estAngleMean` An optional named list indicating whether or not to estimate the angle mean for data streams with angular distributions (e.g. turning angle). If not estimated (the default), the angle mean is fixed to 0.
- `formula` Regression formula for the transition probability covariates
- `stationary` Logical indicating whether or not the initial distribution is considered equal to the stationary distribution (must be `FALSE` if `formula` includes covariates)
- `Par0` A named list containing vectors of starting values for the state-dependent probability distribution parameters of each data stream

These seven arguments are all that are needed in order to fit the HMMs currently supported in `moveHMM` (Michelot *et al.* 2016). For example, here is how the analysis of 15 "wild haggis" tracks described in Michelot *et al.* (2016) would be implemented using `momentuHMM`:

```
library(momentuHMM)
### Load raw data
rawHaggis<-read.csv("rawHaggises.csv")
### Process data
processedHaggis<-prepData(data=rawHaggis,covNames=c("slope","temp"))

### Fit HMM
```

```
# initial step distribution natural scale parameters
stepPar0 <- c(1,5,0.5,3) # (mu_1,mu_2,sd_1,sd_2)
# initial angle distribution natural scale parameters
anglePar0 <- c(0,0,1,8) # (mean_1,mean_2,concentration_1,concentration_2)
fitHaggis <- fitHMM(data = processedHaggis, nbStates = 2,
                    dist = list(step = "gamma", angle = "vm"),
                    Par0 = list(step = stepPar0, angle = anglePar0),
                    formula = ~ slope + I(slope^2),
                    estAngleMean = list(angle=TRUE))
```

Note that many of the arguments in `fitHMM` are lists, with each element of the list corresponding to a data stream. The list names provided in `dist`, `Par0`, and `estAngleMean` (e.g. 'step' and 'angle') must therefore have a corresponding column in `data` with the same name. Additional data streams can be added to the model by simply adding the additional elements to these list arguments (see examples in sections 3.2 and 3.8).

As seen above, the `formula` argument can include many of the functions and operators commonly used to construct terms in R linear model formulas (e.g. `a*b`, `a:b`, `cos(a)`). The `formulaDelta` argument can be similarly used to specify covariate models for the initial distribution. Unique to `momentuHMM`, the `formula` argument can also be used to specify transition probability matrix models that incorporate cyclical patterns (using the `cosinor` special function; see example in section 3.1), splines for explaining other more complicated patterns (e.g., `bs` and `ns` functions in the R base package `splines`), and factor variables (e.g., `formula=~ID` for individual-level effects). By default the `formula` argument applies to all state transition probabilities, but the special functions `state`, `toState`, and `betaCol` allow for state- and parameter-specific formulas to be specified (see examples in sections 3.4 and 3.8). While `betaCol` allows a formula to be specified for a specific transition (e.g. state $3 \rightarrow 1$), `state` and `toState` allow a formula to be specified for all transitions from (e.g. $3 \rightarrow 1$, $3 \rightarrow 2$) and to (e.g. state $1 \rightarrow 3$, $2 \rightarrow 3$) specific states, respectively. The `betaCons` argument allows for equality constraints among any of the transition probability parameters (e.g. $\gamma_{12}^{(t)} = \gamma_{21}^{(t)}$; see example in section 3.8). Specific state transition probabilities can also be fixed to zero (or any other value) using the `fixPar` argument, which can be useful for incorporating more behavioral realism. For example, `fixPar` can be used to prohibit or enforce switching from one particular state to another (possibly as a function

of spatio-temporal covariates).

Similar to the `formula` argument for state transition probability modelling, it is through the `DM` argument of `fitHMM` that models are specified for the state-dependent probability distribution parameters for each data stream. `DM` is a list argument containing an element for each data stream, but each element itself is also a list specifying the design matrix formulas for each parameter. For example, the following fits the exact same wild haggis model as above, but employs a user-specified (intercept-only) design matrix for the step length data stream:

```
stepDM <- list(mean = ~1, sd = ~1)

### Fit HMM using user-specified DM
fitHaggisDM <- fitHMM(data = processedHaggis, nbStates = 2,
                      dist = list(step = "gamma", angle = "vm"),
                      DM = list(step = stepDM),
                      Par0 = list(step = log(stepPar0), angle = anglePar0),
                      formula = ~ slope + I(slope^2),
                      estAngleMean = list(angle=TRUE))
```

Note that when `DM` is specified for a data stream, the initial parameter values (`Par0`) for that data stream now correspond to columns of the resulting design matrix and must be on the working scale instead of the natural scale. In this case, because the log link is used for the natural parameters of the gamma distribution, `Par0$step` was specified on the log scale. The functions `getPar`, `getPar0`, `checkPar0`, and `getParDM` are designed to assist users in the specification of design matrices and corresponding initial values on the working scale for any given model (see package documentation for further details). `DM` formulas are just as flexible as the `formula` argument and, in addition to common linear model formula functions and operators, can also include cyclical cosinor models (see section 3.1), splines, factor variables, and state-specific probability distribution parameter formulas (see examples in sections 3.3 and 3.4). As with the state transition probabilities, working parameters for probability distributions can also be fixed to user-specified values using the `fixPar` argument.

Specification of design matrices using `DM` is not limited to formulas. Alternatively, "pseudo-design" matrices can be specified, using an R matrix with rows corresponding to the natural parameters and columns corresponding to the working parameters. The elements in the matrix may be numeric or character strings containing model formula

terms (see examples in sections 3.4, 3.7, and 3.8). Using a pseudo-design matrix for step length, the following is yet another way to implement the exact same wild haggis model:

```
stepDMp <- matrix(c(1,0,0,0,
                    0,1,0,0,
                    0,0,1,0,
                    0,0,0,1),4,4,byrow=TRUE)
rownames(stepDMp) <- c("mean_1","mean_2","sd_1","sd_2")
colnames(stepDMp) <- c("mean_1:(Intercept)","mean_2:(Intercept)",
                       "sd_1:(Intercept)","sd_2:(Intercept)")

### Fit HMM using user-specified DM
fitHaggisDMp <- fitHMM(data = processedHaggis, nbStates = 2,
                       dist = list(step = "gamma", angle = "vm"),
                       DM = list(step = stepDMp),
                       Par0 = list(step = log(stepPar0), angle = anglePar0),
                       formula = ~ slope + I(slope^2),
                       estAngleMean = list(angle=TRUE))
```

(note that column and rownames for pseudo-design matrices are not required but can be useful). Pseudo-design matrices allow for the sharing of common working parameters (such as intercept terms) among natural scale parameters, and this can be used to constrain natural scale parameters (e.g., $\mu_1 \leq \mu_2$) when used in tandem with the `workBounds` argument (see sections 3.2, 3.7, and 3.8). This is particularly useful for preventing state label switching when repeatedly fitting the same HMM using multiple imputation methods (see section 2.4).

2.3 Circular-circular regression model for the angle mean

Another noteworthy `fitHMM` argument, `circularAngleMean`, is a list argument that enables users to specify circular-circular regression models for the mean (μ) parameter of angular distributions, such as the wrapped Cauchy and von Mises, instead of circular-linear models based on the tangent link function (Table 2). When `circularAngleMean` is specified as `TRUE` for any given angular data stream (e.g. turning angle), then a special link function based on Rivest *et al.* (2016) is used:

$$\boldsymbol{\mu} = \text{atan2}(\sin(\mathbf{X}_\mu)\boldsymbol{\beta}_\mu, 1 + \cos(\mathbf{X}_\mu)\boldsymbol{\beta}_\mu), \tag{2}$$

where $\mathbf{X}_\mu$ is a $T \times K$ matrix composed of the turning angles between K angular covariates (e.g., wind direction, sea surface current direction) and the bearing of movement during the previous time step; that is, each element

$$x_{t,k} = \text{atan2}(\sin(r_{t,k} - b_{t-1}), \cos(r_{t,k} - b_{t-1})) \tag{3}$$

for angular covariate $r_{t,k}$ and $k = 1, \ldots, K$ (note that `prepData` and `MIfitHMM` calculate $\mathbf{X}_\mu$ based on the `angleCovs`, `centers`, or `centroids` arguments so users need not bother). Because this link function is designed for turning angles, a turning angle of 0 is provided as the reference angle (hence the "1+" preceeding the cosine term in Eq. 2). Thus as a trade-off between biased and correlated movements, the working parameters ($\boldsymbol{\beta}_\mu$) for the expected turning angle at time t weight the attractive (or repulsive) strengths of the angular covariates relative to directional persistence. When all $\boldsymbol{\beta}_\mu = 0$, the model reduces to a correlated random walk, but an increasingly biased random walk results as $\boldsymbol{\beta}_\mu$ gets larger (or smaller). Many interesting hypotheses about animal movmement can be addressed using circular-circular regression on movement direction, including the effects of wind, sea surface currents (see example in section 3.3), centers of attraction or repulsion (see examples in sections 3.4 and 3.8), group dynamic models (see example in section 3.6), and dynamic activity centers (see example in section 3.8.

The special function `angleFormula` can be included in `DM` formulas or pseudo-design matrices in order to model the circular-circular regression angle mean as a function of the relative strength (or importance) of angular covariates (Rivest *et al.* 2016):

$$\boldsymbol{\mu} = \text{atan2}((\mathbf{Z}_\mu \circ \sin(\mathbf{X}_\mu))\boldsymbol{\beta}_\mu, 1 + (\mathbf{Z}_\mu \circ \cos(\mathbf{X}_\mu))\boldsymbol{\beta}_\mu), \tag{4}$$

where $\mathbf{Z}_\mu$ is a $T \times K$ matrix of positive real covariates (e.g. wind speed, sea surface current speed) and $\circ$ is the Hadamard (i.e. element-wise) product. The special function `angleFormula` can also be used to specify group- or individual-level effects on the circular-circular regression angle mean coefficients ($\boldsymbol{\beta}_\mu$).

Also based on Rivest *et al.* (2016), the von Mises consensus distribution is a special von Mises circular-circular regression model where the concentration parameter (ρ) depends on the level of agreement among short-term directional persistence (i.e. moving

forward) and the angular covariates:

$$\boldsymbol{\rho} = \kappa\sqrt{\left[(\mathbf{Z}_\mu \circ \sin(\mathbf{X}_\mu))\boldsymbol{\beta}_\mu\right]^2 + \left[1 + (\mathbf{Z}_\mu \circ \cos(\mathbf{X}_\mu))\boldsymbol{\beta}_\mu\right]^2}. \quad (5)$$

Note that the von Mises consensus distribution is parameterized in terms of μ and κ (see Table 2), but `momentuHMM` returns and plots real parameter estimates in terms of μ and ρ. When all $\boldsymbol{\beta}_\mu$ are non-negative, then the minimum and maximum values for ρ are $\kappa|1 - \min(\mathbf{Z}_\mu\boldsymbol{\beta}_\mu)|$ and $\kappa\left[1 + \max(\mathbf{Z}_\mu\boldsymbol{\beta}_\mu)\right]$, respectively. In the consensus model, κ can be interpreted as the concentration towards a turning angle of zero (i.e. moving forward) when the angular covariate components perfectly cancel out. See section 3.3 for example code using `angleFormula` and the von Mises consensus ("vmConsensus") distribution.

2.4 Multiple imputation

When location data are temporally-irregular or subject to measurement error, then they are not suitable for standard maximum-likelihood HMM analyses based on the forward algorithm (Eq. 1). In this case, `momentuHMM` can be used to perform the 2-stage multiple imputation approach of McClintock (2017). The basic concept is to first employ a single-state (i.e., $N = 1$) movement model that is relatively easy to fit but can accommodate location measurement error and temporally-irregular or missing observations (e.g. Johnson *et al.* 2008). The second stage involves repeatedly fitting the desired HMM to m temporally-regular realizations of the position process drawn from the model output of the first stage. Data streams or covariates that are dependent on location (e.g., step length, turning angle, habitat type, snow depth, sea surface temperature) will of course vary among the m realizations of the position process, and the pooled inferences across the HMM analyses therefore reflect location uncertainty.

There are three primary functions (`MIfitHMM`, `MIpool`, and `crawlWrap`) for performing multiple imputation HMM analyses in `momentuHMM`, and all rely on parallel processing to speed up computations. `crawlWrap` is a wrapper function for fitting the continuous-time correlated random walk (CTCRW) model of Johnson *et al.* (2008) to one or more tracks (subject to location measurement error and/or temporal irregularity) and then predicting temporally-regular tracks of the user's choosing (e.g. 15 min, hourly, daily) based on the CTCRW model output. `crawlWrap` returns a `crwData` ob-

ject that can be used to draw m realization of the position process within the `MIfitHMM` function. `MIfitHMM` is essentially a wrapper function for `fitHMM` that repeatedly fits the same user-specified HMM to m imputed data sets and stores the output from each of the m model fits. If a `crwData` object is provided, then `MIfitHMM` will first draw m imputations based on the `crwData` output and then fit the specified HMM to each imputed data set. If users wish to use a movement model other than the CTCRW to account for measurement error and temporal irregularity (e.g. Calabrese *et al.* 2016; Gurarie *et al.* 2017), or if other observation error processes (e.g. missing data) are to be accounted for in the imputation step, `MIfitHMM` can also be used for analysis of a list of m `momentuHMMData` objects that were imputed by the user. Based on the m model fits, the `MIpool` function calculates pooled estimates, standard errors, and confidence intervals for the working scale parameters, natural scale parameters (based on transformations of the pooled working parameters and mean or user-specified values for any covariates), state sequences, state probabilities, and activity budgets (i.e. the proportion of the T times step assigned to each state) using standard multiple imputation formulae (Rubin & Schenker 1986; McClintock 2017). `MIpool` can be called separately or within `MIfitHMM` (using the `poolEstimates` argument), and the function returns a `miSum` object containing the pooled output across all imputatons. See sections 3.2, 3.3, 3.4, and 3.7 for example HMM analyses that use multiple imputation to account for location measurement error and temporally irregularity.

2.5 Model visualization and diagnostics

The generic `plot` functions for `momentuHMM` models (`plot.momentuHMM` and `plot.miSum`) plot the data stream histograms along with their corresponding estimated probability distributions, the estimated natural parameters and state transition probabilities as a function of any covariates included in the model, and the tracks of all individuals (color-coded by the most likely state sequence). By default, the probability distributions are plotted based on the means of any covariate values, but user-specified covariate values for the plots can be provided using the `covs` argument. When the argument `plotCI=TRUE`, then confidence intervals for the natural parameters and state transition probabilities are also plotted. Confidence intervals are calculated from the working parameter estimates based on the delta method and finite-difference approximations of the first derivative for the transformation using the `numDeriv::grad` function (Gilbert

& Varadhan 2016). For multiple imputation analyses (`plot.miSum`), all plots are based on the pooled parameter estimates and the means of any covariates (if not provided by the `covs` argument) across each imputation. Using the argument `errorEllipse`, `plot.miSum` will include estimated location error ellipses in the plots of individual tracks. The functions `plotSat`, `plotSpatialCov`, and `plotStates` (Table 1) provide further methods for visualizing model results.

Diagnostic tools include the calculation and plotting of pseudo-residuals (Zucchini *et al.* 2016) using the `pseudoRes` and `plotPR` functions, respectively. For discrete distributions (e.g. Bernoulli, Poisson), a continuity adjustment is used for calculating pseudo-residuals. Akaike's Information Criterion can be calculated for one or more models using the `AIC.momentuHMM` function.

2.6 Simulation

The function `simData` can be used to simulate multivariate HMM data from scratch or based on the estimated parameters of existing `momentuHMM` or `miSum` models. The `simData` arguments are very similar to those used for model specification in `fitHMM` (e.g., `dist`, `DM`) and data preparation in `prepData` (e.g., `spatialCovs`, `centers`), but they include additonal arguments, `lambda` and `errorEllipse`, for simulating location data subject to temporal irregularity and measurement error, respectively. The `spatialCovs` argument allows for rasters of spatio-temporal covariate values to be utilized in simulation models, while the `centers` argument allows activity centers to be incorporated. Thus `simData` can be used to simulate more ecologically-realistic tracks (potentially subject to observation error) that can be useful for study design, power analyses, and assessing model performance. Goodness-of-fit can also be investigated by drawing simulated data sets from a fitted model and comparing them to observed properties of the data (Morales *et al.* 2004).

3 Examples

We will now demonstrate some of the capabilities of `momentuHMM` using real telemetry data. These examples are intended for demonstration purposes only, and we do not claim these example analyses represent improvements relative to previous or alternative analyses for these data sets. While only some of the key workflow elements are

included here, complete R code and further details for these analyses are available in the "vignettes" source directory.

3.1 African elephant

As our first example, we use an African elephant (*Loxodonta africana*) bull track described in Wall *et al.* (2014) and publicly available from the movebank.org data repository.

We can load the data from the URL (this requires an Internet connection):

```
URL <- paste0("https://www.datarepository.movebank.org/bitstream/handle/",
              "10255/move.373/Elliptical%20Time-Density%20Model%20%28Wall%",
              "20et%20al.%202014%29%20African%20Elephant%20Dataset%20%",
              "28Source-Save%20the%20Elephants%29.csv")
rawData <- read.csv(url(URL))
```

The data set contains two tracks; for this analysis, we only consider the first one. In addition to hourly locations, the tag also collected external temperature data. We subset the data frame to keep only the relevant rows and columns:

```
# select and rename relevant columns
rawData <- rawData[,c(11,3,4,5,6)]
colnames(rawData) <- c("ID","time","lon","lat","temp")

# only keep first track
rawData <- subset(rawData,ID==unique(ID)[1])
```

The data now has the following columns:

```
head(rawData)
```

```
##            ID                time       lon      lat temp
## 1 Salif Keita 2008-03-22 17:00:00.000 -2.160167 15.65350   38
## 2 Salif Keita 2008-03-22 18:00:00.000 -2.160075 15.65452   35
## 3 Salif Keita 2008-03-22 19:00:00.000 -2.159902 15.65451   32
## 4 Salif Keita 2008-03-22 20:00:00.000 -2.159435 15.65489   30
## 5 Salif Keita 2008-03-22 21:00:00.000 -2.158113 15.65512   29
## 6 Salif Keita 2008-03-22 22:00:00.000 -2.157848 15.65461   28
```

Location measurement error is negligible for these terrestrial GPS data, although about 1% of the hourly observations collected between 22 March 2008 and 30 September 2010 are missing. Instead of simply ignoring these missing data, we can employ `crawlWrap` to predict the missing locations based on the CTCRW model of Johnson *et al.* (2008) prior to conducting our HMM analysis.

To use `crawlWrap`, we convert times from factors to POSIX, and project the observed locations to UTM coordinates:

```
# convert times from factors to POSIX
rawData$time <- as.POSIXct(rawData$time,tz="GMT")

# project to UTM coordinates using package rgdal
library(rgdal)
llcoord <- SpatialPoints(rawData[,3:4],
                         proj4string=CRS("+proj=longlat +datum=WGS84"))
utmcoord <- spTransform(llcoord,CRS("+proj=utm +zone=30 ellps=WGS84"))

# add UTM locations to data frame
rawData$x <- attr(utmcoord,"coords")[,1]
rawData$y <- attr(utmcoord,"coords")[,2]
```

Then, we call `crawlWrap` to fit a CTCRW model and predict hourly locations:

```
# fit crawl model
crwOut <- crawlWrap(obsData=rawData, timeStep="hour",
                    theta=c(6.855, -0.007), fixPar=c(NA,NA))
```

Here the desired time step is specified by the `timeStep` argument, and `theta` and `fixPar` arguments are the same as for `crawl::crwMLE` (Johnson 2017). For the choice of initial parameters in `crawlWrap`, we refer the reader to the documentation of the package crawl, in particular `crawl::crwMLE` and `crawl::crwPredict`. We now have a complete set of temporally-regular location data.

Autocorrelation function (ACF) estimates suggest there are 24-hour cycles in the step length data, and this presents an opportunity to demonstrate the use of the `cosinor` function for incorporating cyclical behavior in model parameters using `momentuHMM`. We create a `momentuHMMData` object, and the 24-hour cosinor model covariate:

```
# create momentuHMMData object from crwData object
elephantData <- prepData(data=crwOut, covNames="temp")

# add cosinor covariate based on hour of day
elephantData$hour <- as.integer(strftime(elephantData$time, format = "%H", tz="GMT"))
```

As seen here, the function `prepData` can also be used for pre-processing the best predicted track data from `crawlWrap` output. The 24-hour cosinor covariate ("hour") is simply a set of integers $(0, 1, \ldots, 23)$ indicating the hour of day for each observation. The ACF plot of the step lengths, shown in Figure 1, was obtained with:

```
acf(elephantData$step[!is.na(elephantData$step)],lag.max=300)
```

Our aim is to fit a 2-state HMM to the elephant track that includes temperature effects on the turning angle concentration parameters and cycling temperature effects (with a 24-hour periodicity) on the step length and state transition probability parameters. Complex models such as this can require many parameters, and it can be challenging to choose good starting parameter values for the optimization. Here, we take an incremental approach, starting from a simpler model with no covariates. In `momentuHMM`, the function `getPar0` extracts initial parameters from a fitted (nested) HMM, given arguments for the more complex model.

For the covariate-free 2-state model, six initial parameters need to be chosen: for each state, the mean and standard deviation of the gamma distribution of step lengths, and the concentration of the wrapped Cauchy distribution of turning angles. Looking at the histograms of the step lengths and the turning angles (e.g. output by `plot(elephantData)`) is often useful to choose good starting parameter values.

```
# label states
stateNames <- c("encamped","exploratory")
# distributions for observation processes
dist = list(step = "gamma", angle = "wrpcauchy")

# initial parameters
Par0_m1 <- list(step=c(100,500,100,200),angle=c(0.3,0.7))

# fit model
m1 <- fitHMM(data = elephantData, nbStates = 2, dist = dist, Par0 = Par0_m1,
```

```
            estAngleMean = list(angle=FALSE), stateNames = stateNames)
```

To ensure convergence, we could also use the argument `retryFits` to specify the number of attempts to minimize the negative log-likelihood based on random perturbations of the parameter estimates at the current minimum.

We can build on complexity, by including the temperature and time of day as covariates in the state transition probabilities. We use the function `getPar0` to extract the new starting parameter values.

```
# formula for transition probabilities
formula <- ~ temp * cosinor(hour, period = 24)

# initial parameters (obtained from nested model m1)
Par0_m2 <- getPar0(model=m1, formula=formula)

# fit model
m2 <- fitHMM(data = elephantData, nbStates = 2, dist = dist, Par0 = Par0_m2$Par,
             beta0=Par0_m2$beta, stateNames = stateNames, formula=formula)
```

The special function `cosinor(hour, period = 24)` internally creates the cosinor model covariates, $\cos(2\pi \times hour/period)$ and $\sin(2\pi \times hour/period)$, and includes both terms (plus interactions with "temp") in the fitted model.

Finally, we can fit the more complex model, including the effect of temperature and time of day on the parameters of the state-dependent distributions of steps and angles.

```
# formulas for parameters of state-dependent observation distributions
DM <- list(step = list(mean = ~ temp * cosinor(hour, period = 24),
                       sd = ~ temp * cosinor(hour, period = 24)),
           angle = list(concentration = ~ temp))

# initial parameters (obtained from nested model m2)
Par0_m3 <- getPar0(model=m2, formula=formula, DM=DM)

# fit model
m3 <- fitHMM(data = elephantData, nbStates = 2, dist = dist, Par0 = Par0_m3$Par,
             beta0 = Par0_m3$beta, DM = DM, stateNames = stateNames,
             formula = formula)
```

The above model `m3` identifed a state of slow undirected movement ("encamped"), and a state of faster and more directed movement ("exploratory") (Figure 1). For a fitted model, the function `viterbi` computes the most likely state sequence:

```
# decode most likely state sequence
states <- viterbi(m3)
# derive percentage of time spent in each state
table(states)/nrow(elephantData)
```

Here, about 74% of the steps were attributed to the "encamped" state, and 26% were attributed to the "exploratory" state.

We can use `AIC(m1,m2,m3)` to compare the three fitted models in terms of AIC; here, `m3` is overwhelmingly supported by the AIC when compared to alternative models with fewer covariates.

The model can be visualized with the generic function `plot`, which was used for the plots shown in Figure 2, and the decoded track in Figure 1.

```
plot(m3, plotCI = TRUE, covs = data.frame(hour=12))
```

Interestingly, this model suggests step lengths and directional persistence for the "encamped" state decreased as temperature increased, step lengths for both states tended to decrease in the late evening and early morning, and transition probabilities from the "encamped" to "exploratory" state decreased as temperature increased (Figure 2).

Model fit can be assessed using the pseudo-residuals, with the functions `pseudoRes` and `plotPR`. The residual ACF plot shown in Figure 1 was produced by:

```
# compute pseudo-residuals for the steps and the angles
pr <- pseudoRes(m3)

# plot the ACF of step pseudo-residuals
acf(pr$stepRes[!is.na(pr$stepRes)],lag.max = 300)
```

Autocorrelation function plots of the pseudo-residuals indicate this model explained much of the periodicity in step length, although there does still appear to be some room for improvement.

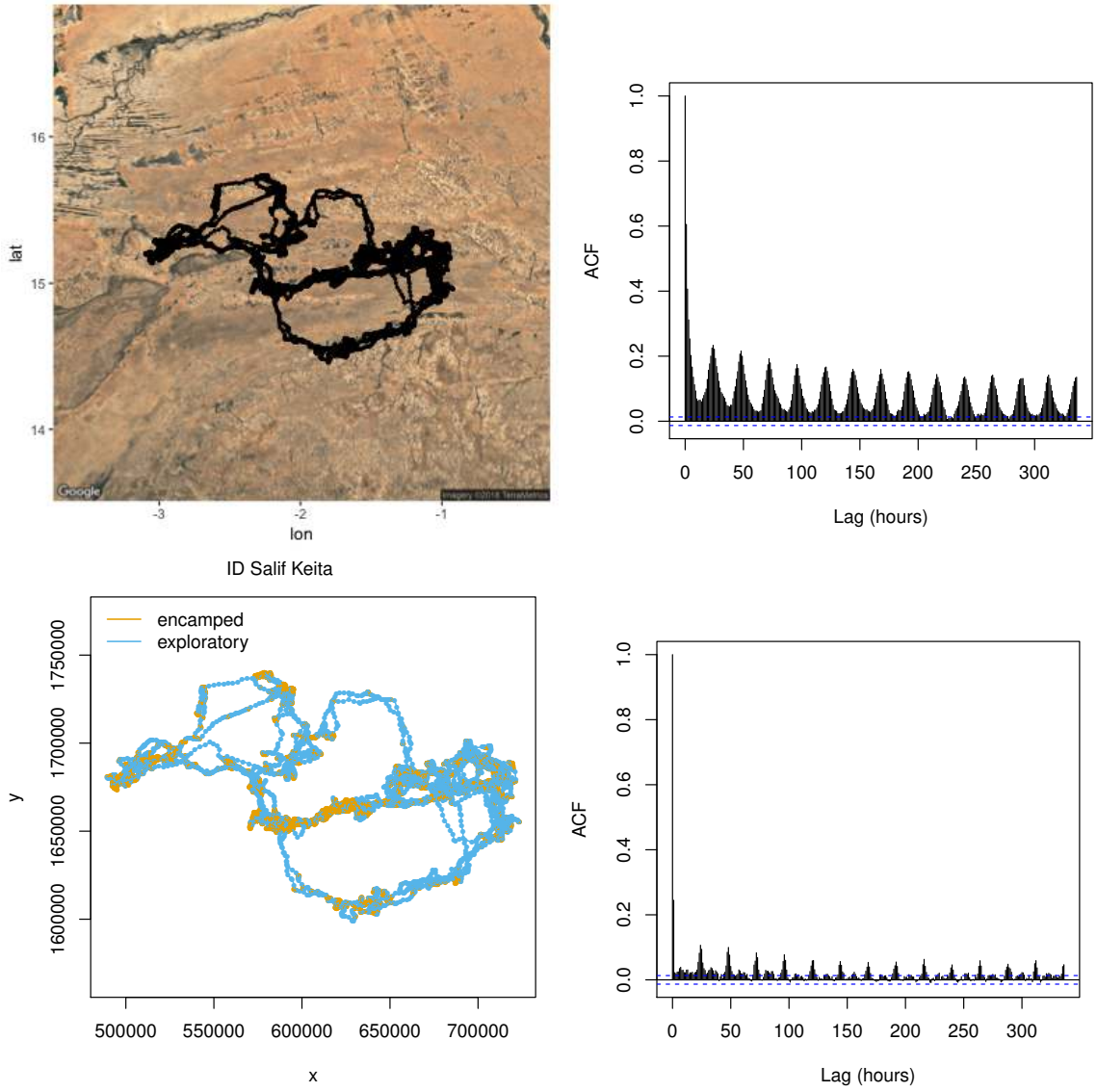

Figure 1. *Plot of the elephant track produced using the 'plotSat' function (top-left panel), autocorrelation function (ACF) plot of the corresponding step length data (top-right panel), plot of the Viterbi-decoded state sequence for the 2-state ("encamped" and "exploratory") model generated using the generic 'plot' function (bottom-left panel), and the step length pseudo-residual ACF plot for this model using the 'plotPR' function (bottom-right panel).*

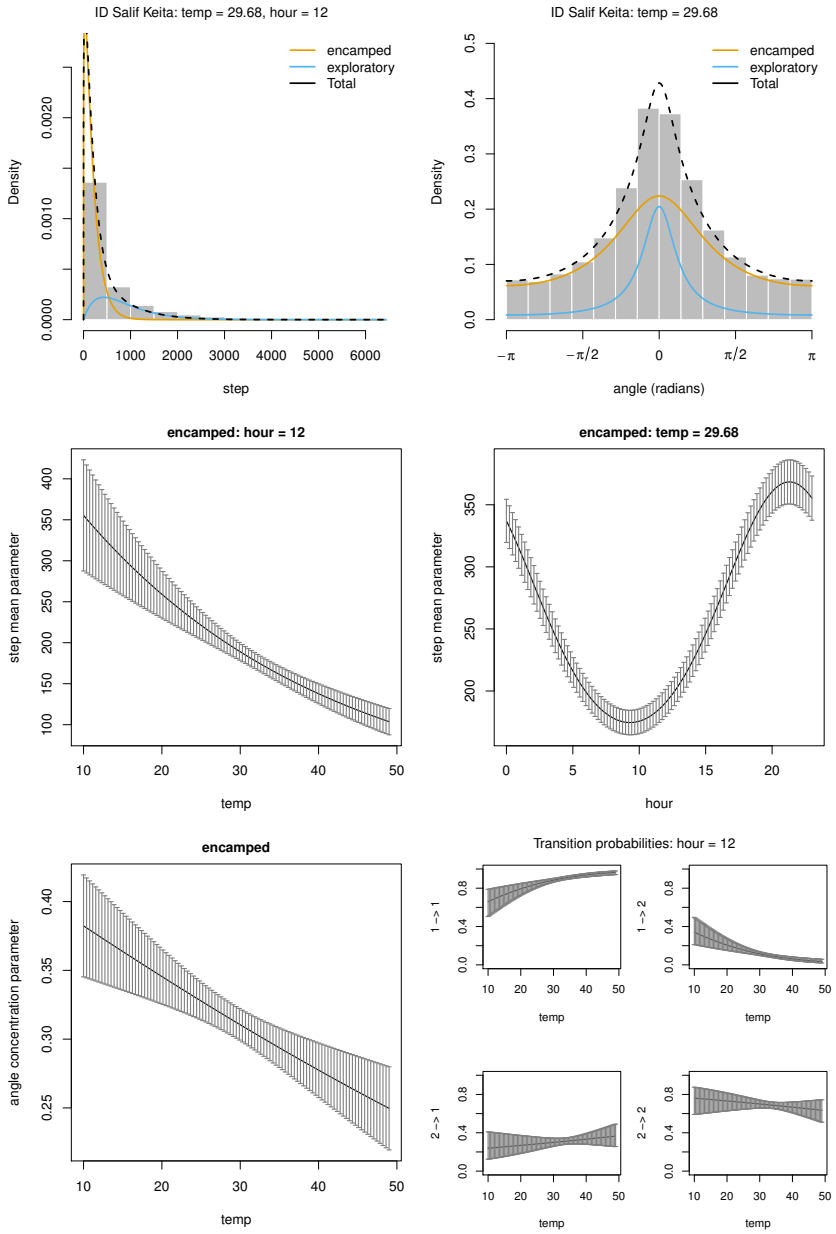

Figure 2. *Selected plots for the 2-state ("encamped" and "exploratory") African elephant example generated using the generic 'plot' function. Top panels present histograms of the step length (top-left) and turning angle (top-right) data along with the estimated state-dependent probability distributions based on the mean temperature (temp = 29.7 degrees celsius) at 12:00 GMT (hour = 12). Middle panels present estimates (and 95% confidence intervals) for the step length mean parameter of the "encamped" state as a function of temperature and hour of day. Bottom-left panel presents estimates for the turning angle concentration parameter of the "encamped" state as a function of temperature. Bottom-right panel presents estimated state transition probabilities (1 = "encamped", 2 = "exploratory") as a function of temperature at 12:00 GMT.*

3.2 Northern fur seal

In our second example, we use the northern fur seal (*Callorhinus ursinus*) example from McClintock *et al.* (2014) to demonstrate the use of additional data streams for distinguishing behaviors with similar horizontal trajectories in a multivariate HMM. The data consist of 241 temporally-irregular Fastloc GPS locations obtained during a foraging trip of a nursing female near the Pribilof Islands of Alaska, USA, from 10-17 October 2007. The tag included time-depth recording capabilities, and the dive activity data were summarized as the number of foraging dives over $T = 228$ temporally-regular 1 hr time steps. To fit the $N = 3$ state (1="resting", 2="foraging", 3="transit") of McClintock *et al.* (2014) using `momentuHMM`, we first used `crawlWrap` to predict temporally-regular locations at 1 hr time steps assuming a bivariate normal measurement error model and merged the results with the foraging dive data using the `crawlMerge` function. Then multiple imputation was used to account for location measurement error by repeatedly fitting the HMM to `nSims` realizations of the position process using `MIfitHMM`:

```
nbStates <- 3
stateNames <- c("resting", "foraging", "transit")
dist <- list(step = "gamma", angle = "wrpcauchy", dive = "pois")
Par0 <- getParDM(nbStates = nbStates, dist = dist,
                 Par = Par, DM = DM, workBounds = workBounds,
                 estAngleMean = list(angle = FALSE))
Fixpar <- list(dive = c(-100, NA, NA))
nfsFits <- MIfitHMM(crwOut, nSims = 100, nbStates = nbStates, dist = dist,
                    Par0 = Par0, DM = DM, workBounds = workBounds,
                    estAngleMean = list(angle = FALSE),
                    fixPar = fixPar, retryFits = 30,
                    stateNames=stateNames)
plot(nfsFits)
```

Here we specified a gamma distribution for step length ('step'), wrapped Cauchy distribution for turning angle ('angle'), and Poisson distribution for the number of foraging dives ('dive'). The function `getParDM` was used to organize the starting values for the data stream working parameters (`Par0`) in the correct format based on `DM`, `workBounds`, and estimates of the natural parameters (`Par`) from McClintock *et al.* (2014). The `DM` and `workBounds` arguments were specified to avoid label switching among the `nSims` imputed data model fits and enforce similar state-dependent probability distribution

constraints as McClintock *et al.* (2014); for example, constraining the Poisson rate parameters such that the "foraging" state tends to have higher numbers of foraging dives than the "transit" state ($\lambda_2 > \lambda_3$; see Eq. 10 in section 3.7 for more details on parameter constraints using `DM` in conjunction with the `userBounds` and `workBounds` arguments). To prohibit foraging dives for the "resting" state, we used the `fixPar` argument to effectively fix the Poisson rate parameter to zero on the natural scale (i.e. $\lambda_1 \approx 0$). To help deal with the problem of convergence to local maxima, the `retryFits` argument allows users to specify the number of times to attempt to re-fit each model using random perturbations of the parameter estimates as the starting values for optimization.

The results are very similar to those of the discrete-time model of McClintock *et al.* (2014), with periods of foraging often followed by resting (Figure 3). The "activity budgets" (i.e. the proportion of time steps allocated to each state) calculated by `MIpool` based on the estimated state sequences for each imputation were 0.31 (95% CI: 0.25−0.37) for "resting", 0.28 (95% CI: 0.23−0.35) for "foraging", and 0.41 (95% CI: 0.33−0.5) for "transit".

3.3 Loggerhead turtle

For our third example, we demonstrate how to model movement direction and step length as a function of angular covariates using hitherto unpublished loggerhead turtle (*Caretta caretta*) data for a captive-raised juvenile released in 2012 on the coast of North Carolina, USA. The data consist of 165 temporally-irregular Argos locations subject to measurement error and rasters of daily ocean surface currents collected between 20 November and 19 December 2012. Assuming a gamma distribution for step length (l_t) and a wrapped Cauchy distribution for turning angle (ϕ_t), we model the mean step length parameter (μ_t^l) as a function of ocean surface current speed (w_t) and direction (r_t) relative to the bearing of movement (b_t):

$$\mu_t^l = \exp(\beta_0^l + \beta_1^l w_t \cos(b_t - r_t)), \tag{6}$$

and the turning angle mean parameter (μ_t^ϕ) as a trade-off between short-term directional persistence and bias in the direction of ocean surface currents using the circular-circular regression link function:

$$\mu_t^\phi = \text{atan2}(\sin(d_t)\beta^\phi, 1 + \cos(d_t)\beta^\phi), \tag{7}$$

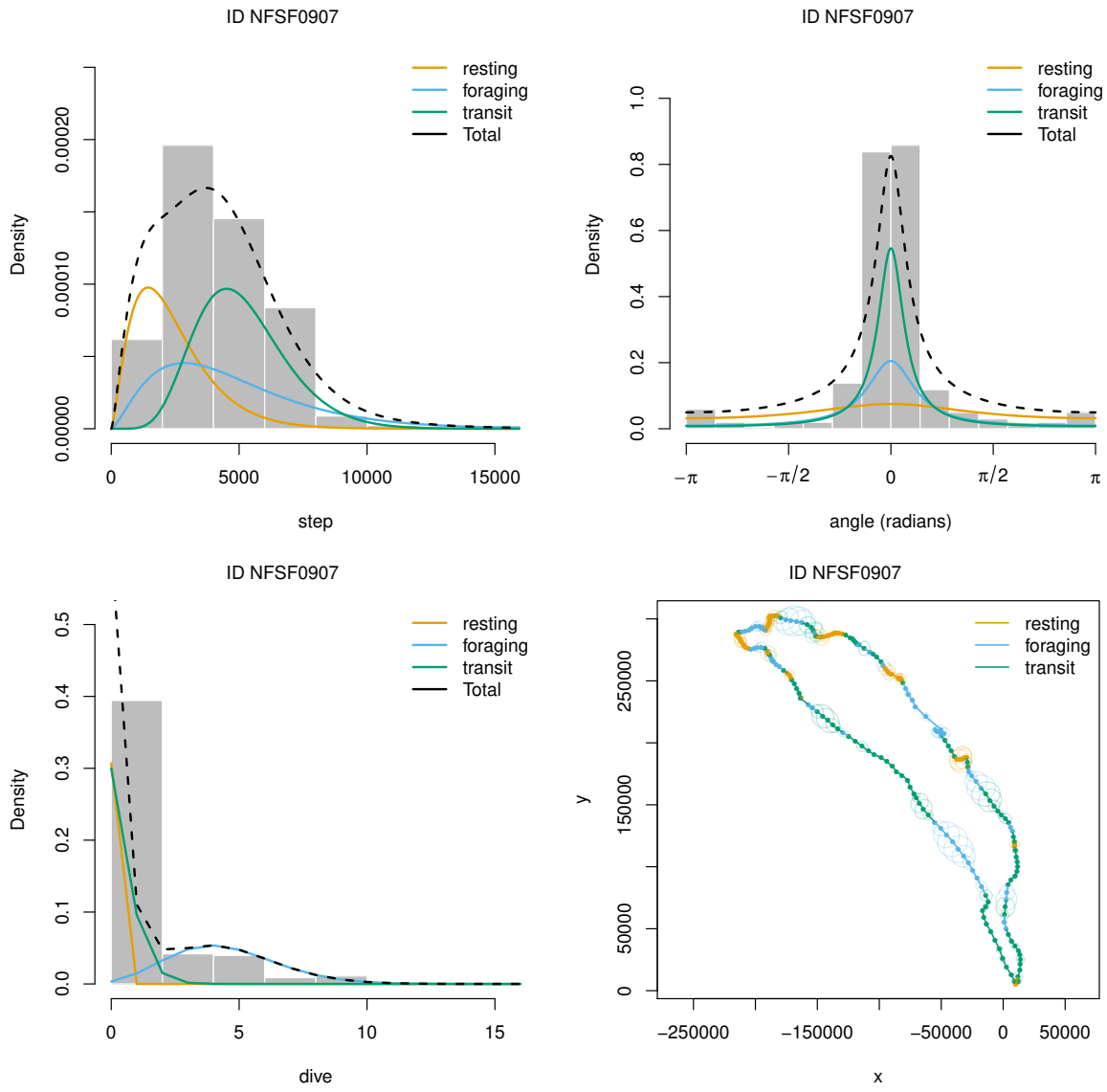

Figure 3. *Plots of the northern fur seal example results generated using the generic 'plot' function. The estimated probability distributions for step length (top-left panel), turning angle (top-right panel), and number of foraging dives (bottom-left panel) for the 3-state ("resting", "foraging", and "transit") model are plotted along with histograms of these data streams. The temporally-regular predicted locations (and 95% ellipsoidal confidence bands) and estimated states are plotted in the bottom-right panel. All estimates are pooled across multiple imputations of the position process and thus reflect uncertainty attributable to location measurement error and temporally-irregular observations.*

where $d_t = \text{atan2}(\sin(r_t - b_{t-1}), \cos(r_t - b_{t-1}))$.

We wish to fit a 2-state HMM to the turtle data, with a "foraging" state unaffected by currents and a "transit" state potentially influenced by ocean surface currents as in Eqs. 6 and 7. We used `crawlWrap` to predict $T = 350$ temporally-regular locations at 2 hr time steps assuming a bivariate normal measurement error model that accounts for the Argos location quality class (i.e. 3,2,1,0,A,B) of each observation. We then again used multiple imputation to account for location uncertainty by repeatedly fitting the HMM to `nSims` realizations of the position process using `MIfitHMM`. We first draw `nSims` realizations of the position process and extract the corresponding spatial covariates from the raster bricks for ocean surface current speed ("speedBrick") and direction ("dirBrick") using `MIfitHMM` with `fit=FALSE`:

```
miTurtleData <- MIfitHMM(crwOut, nSims = 100, fit=FALSE,
                  spatialCovs = list(w = speedBrick, d = dirBrick, r = dirBrick),
                  angleCovs = "d")
```

When the `fit` argument is `FALSE`, `MIfitHMM` returns a list of length `nSims` composed of `momentuHMMData` objects (`miData`). For convenience and ease of interpretation, we manually added an additional covariate ($angle\_osc = \cos(b_t - r_t)$) to each of the imputed data sets and fitted the 2-state HMM using Eqs. 6 and 7 for state 2 ("transit"):

```
nbStates<-2
dist <- list(step = "gamma", angle = "wrpcauchy")
DM <- list(step = list(mean = ~state2(w:angle_osc), sd = ~1),
           angle = list(mean = ~state2(d), concentration= ~1))
turtleFits <- MIfitHMM(miTurtleData$miData, nbStates = nbStates, dist = dist,
                       Par0 = Par0, DM = DM,
                       estAngleMean = list(angle = TRUE),
                       circularAngleMean = list(angle = TRUE))
plot(turtleFits, plotCI = TRUE, covs = data.frame(angle_osc = cos(0)))
```

Note that the `state2` special function in `DM` indicates the covariate formulas are specific to state 2 ("transit") and the `circularAngleMean` argument indicates that circular-circular regression link function is to be used on the mean turning angle parameter as in Eq. 7.

For the "transit" state, pooled parameter estimates indicated step lengths increased with ocean surface current speed and as the bearing of movement aligned with ocean surface current direction ($\beta_1^l = 0.4, 95\%$ CI: $0.09 - 0.7$; Figure 4). The estimated wrapped Cauchy distribution for turning angle had mean angles (μ_t^ϕ) biased towards the direction of ocean surface currents for each time step ($\beta^\phi = 0.26, 95\%$ CI: $0.04 - 0.48$), with concentration parameter $\rho_2^\phi = 0.86$ (95% CI: 0.82−0.9) indicating turning angles were concentrated at μ_t^ϕ. Thus movement during the "transit" state appears to strongly follow ocean surface currents (mean $angle\_osc = 0.88, SD = 0.22$), while movement during the "foraging" state exhibited shorter step lengths ($\mu_1^l = 3001\text{m}, 95\%$ CI: $2439 - 3563$) perpendicular to ocean surface currents (mean $angle\_osc = 0.07, SD = 0.27$), with some directional persistence ($\rho_1^\phi = 0.49, 95\%$ CI: $0.37 - 0.61$). The turtle spent 0.56 (95% CI: 0.46−0.66) of the 2 hr time steps in the "foraging" state and 0.44 (95% CI: 0.34−0.54) of time steps in the "transit" state as it travelled northeast along a predominant current until it (presumably) found an attractive foraging patch (Figure 4).

It may often make more sense to weight angular covariates (such as ocean surface current direction) by their relative strength or importance. For example, weak ocean surface currents may be less likely to influence movement direction than strong ocean surface currents. This could easily be included in our turtle model using the `angleFormula(cov, strength, by)` special function in `DM`, where `cov` is an angle covariate (e.g. wind direction), `strength` is an optional positive real covariate (e.g. wind speed), and `by` is an optional factor variable for individual- or group-level effects (e.g. ID, sex):

```
DM$angle = list(mean = ~state2(angleFormula(d, strength = w)),
                concentration= ~1))
```

which would yield the following model for the "transit" state mean angle parameter:

$$\mu_t^\phi = \text{atan2}(w_t \sin(d_t)\beta^\phi, 1 + w_t \cos(d_t)\beta^\phi). \tag{8}$$

Still another option would be to use the von Mises consensus model, where the concentration parameter would now depend on the level of agreement between short-term directional persistence (i.e. going forward) and ocean surface currents:

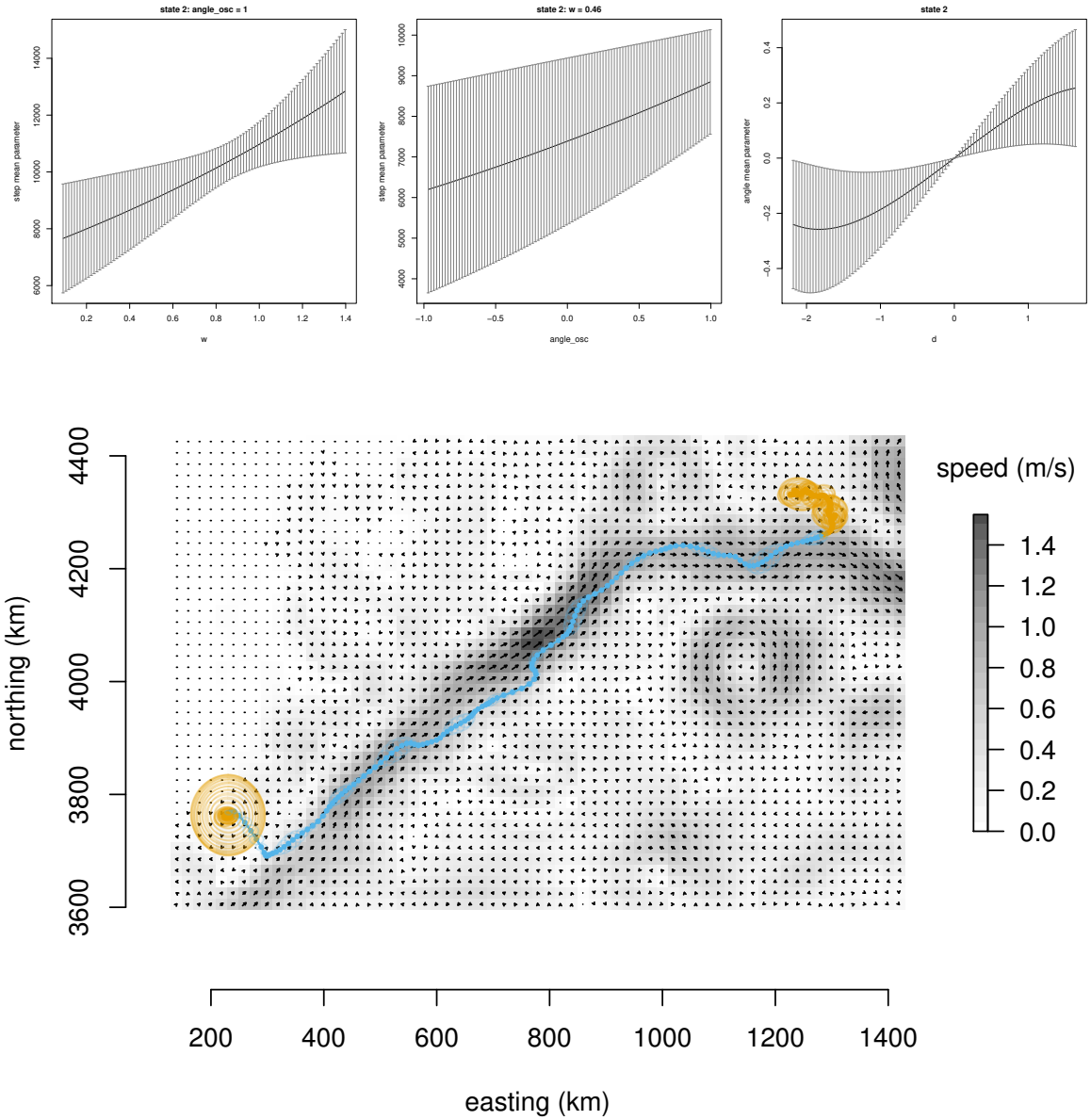

Figure 4. *Selected results from the loggerhead turtle example. Top panels include estimates and 95% confidence intervals for the mean step length parameter as a function of ocean surface current speed (w) when ocean surface current direction (r_t) is the same as the bearing (b_t) of movement (i.e. $angle\_osc = \cos(b_t - r_t) = 1$; top-left panel), mean step length parameter as a function of $angle\_osc$ at the mean ocean surface current speed ($w = 0.46$ m/s; top-middle panel), and mean turning angle parameter as a function of $d_t = atan2(\sin(r_t - b_{t-1}), \cos(r_t - b_{t-1}))$ (top-right panel). Bottom panel plots the pooled track, 95% error ellipse confidence bands, and state (orange = "foraging", blue = "transit") estimates based on multiple imputations of the position process relative to ocean surface current speed (m/s) and direction on 2 December 2012.*

```
dist$angle = "vmConsensus"
DM$angle = list(mean = ~state2(angleFormula(d, strength = w)),
                kappa = ~1)
```

which would yield the following model for the "transit" state concentration parameter:

$$\rho_t^{\phi} = \kappa\sqrt{[w_t \sin(d_t)\beta^{\phi}]^2 + [1 + w_t \cos(d_t)\beta^{\phi}]^2}. \tag{9}$$

If there were multiple turtles in this dataset, then individual-level effects could be included on μ^{ϕ} by simply specifying `angleFormula(d, strength = w, by = ID)` or `angleFormula(d, by = ID)` (with no strength effects).

3.4 Grey seal

For our next example, we perform a similar analysis of a grey seal (*Halichoerus grypus*) track that was originally conducted by McClintock *et al.* (2012) using Bayesian methods and (computationally-intensive) Markov chain Monte Carlo. The data consist of 1045 temporally-irregular Fastloc GPS locations collected in the North Sea between 9 April and 11 August 2008. Because the seal repeatedly visited the same haul-out and foraging locations, it provides a nice example for demonstrating how to implement biased movements relative to activity centers using `momentuHMM`. McClintock *et al.* (2012) fitted a 5-state model to these data that included three center of attraction states, with movement biased towards two haul-out sites ("Abertay" and "Farne Islands") and a foraging area ("Dogger Bank"), and two "exploratory" states ("low speed", "high speed") that were unassociated with an activity center. After using `crawlWrap` to predict $T = 1515$ temporally-regular locations at 2 hr time steps including a bivariate normal measurement error model, we can perform a very similar analysis to McClintock *et al.* (2012) in `momentuHMM` by using the `centers` argument and state-specific functions for the probability distribution parameters. A cluster analysis on the observed locations using the R package `dtwclust` (Sarda-Espinosa 2017) identified three centroids with coordinates that were nearly identical to the three activity centers ("Abertay", "Farne Islands", and "Dogger Bank") identified by McClintock *et al.* (2012). We use these coordinates to derive covariates relative to the activity centers when drawing `nSims` realizations of the position process:

```
crwSim <- MIfitHMM(crwOut, nSims = 100, fit=FALSE,
                   center = centers)
```

Specifying the `centers` argument results in the calculation of two covariates for each activity center: the distance (with '.dist' suffix) and angle (with '.angle' suffix) from each location at time t. These covariates can then be used to model parameters as a function of the distance and angle to activity centers for each time step:

```
dist <- list(step = "weibull", angle = "wrpcauchy")
distFormula <- ~state1(I(Abertay.dist>2500)) + state2(I(Farne.dist>2500))
                  + state3(I(Dogger.dist>15000))
angleFormula <- ~state1(Abertay.angle) + state2(Farne.angle)
                  + state3(Dogger.angle)
stepDM <- list(shape = distFormula, scale = distFormula)
angleDM <- list(mean = angleFormula, concentration = distFormula)
DM <- list(step = stepDM, angle = angleDM)
```

Similar to McClintock *et al.* (2012), we assume a Weibull distribution for step length where both the shape and scale parameter depend on the distance from the location at time t to each activity center. For the activity centers on land ("Abertay" and "Farne"), we allow the (state-dependent) step length parameters to change when the seal is beyond 2500m of the haulout. For the "Dogger" activity center, we allow the parameters to change when the seal is beyond 15000m of this (presumably) foraging area. We thus allow the movement behavior to change within these activity center states upon entering or leaving the vicinity of these sites. We assume a wrapped Cauchy distribution for turning angle with (state-dependent) mean angle derived from the direction to each activity center at time t, and the concentration parameter is modeled similarly to the step length parameters. For the two "exploratory" states, we assumed they are simple random walks unaffected by proximity to activity centers. To complete our model specification, we use the `knownStates` argument to assign the seal to the corresponding activity center state whenever it was within the 2500m (haul-out area) or 15000m (foraging area) thresholds for each imputed data set:

```
greySealFits <- MIfitHMM(miDat, nSims = 400,
                         nbStates = 5, dist = dist,
                         Par0 = Par0, beta0 = beta0, fixPar = fixPar,
```

```
                          formula = distFormula,
                          estAngleMean = list(angle=TRUE),
                          circularAngleMean = list(angle=TRUE),
                          DM = DM, knownStates = knownStates)
plot(greySealFits, plotCI = TRUE)
```

As with the step length and turning angle concentration parameters, the state transition probabilities are also allowed to change as a function of distance to activity centers (as specified by the `formula` argument). The starting values (`Par0` and `beta0`) for each imputation were extracted from a single HMM fitted to the best predicted locations from `crawlWrap`, and `fixPar` was used to remove short-term directional persistence (and thus formulate the model as a mixture of biased and simple random walks).

Estimated activity budgets for the 5 states of this multiple imputation HMM were 0.28 $(0.27 - 0.3)$ for the "Abertay" haul-out state, 0.12 $(0.11 - 0.14)$ for the "Farne Islands" haul-out state, 0.37 $(0.35 - 0.38)$ for the "Dogger Bank" foraging state, 0.1 $(0.05 - 0.21)$ for a low-speed "exploratory" state, and 0.12 $(0.07 - 0.22)$ for a high-speed "exploratory" state. All three activity center states exhibited shorter step lengths and less biased movements when within the vicinity of these targets (Figure 5). Results from this analysis were thus very similar to those of McClintock *et al.* (2012), but this implementation required far less computation time and no custom model-fitting algorithms.

The `simData` function can be used to simulate tracks from a fitted model:

```
greySealSim<-simData(model = greySealFits, centers = centers,
                     initialPosition = centers[1,],
                     obsPerAnimal = 1515)
```

A simulated track is presented along with the fitted track in Figure 6. While potentially useful for study design, power analysis, and prediction, the `simData` function can also be helpful in assessing goodness of fit by repeatedly drawing simulated data sets from a fitted model and comparing them to observed properties of the data (e.g. Morales *et al.* 2004).

3.5 Southern elephant seal

Here, we analyse the southern elephant seal (*Mirounga leonina*) data from Michelot *et al.* (2017) using `momentuHMM`. The data set consists of 15 tracks, each encompassing

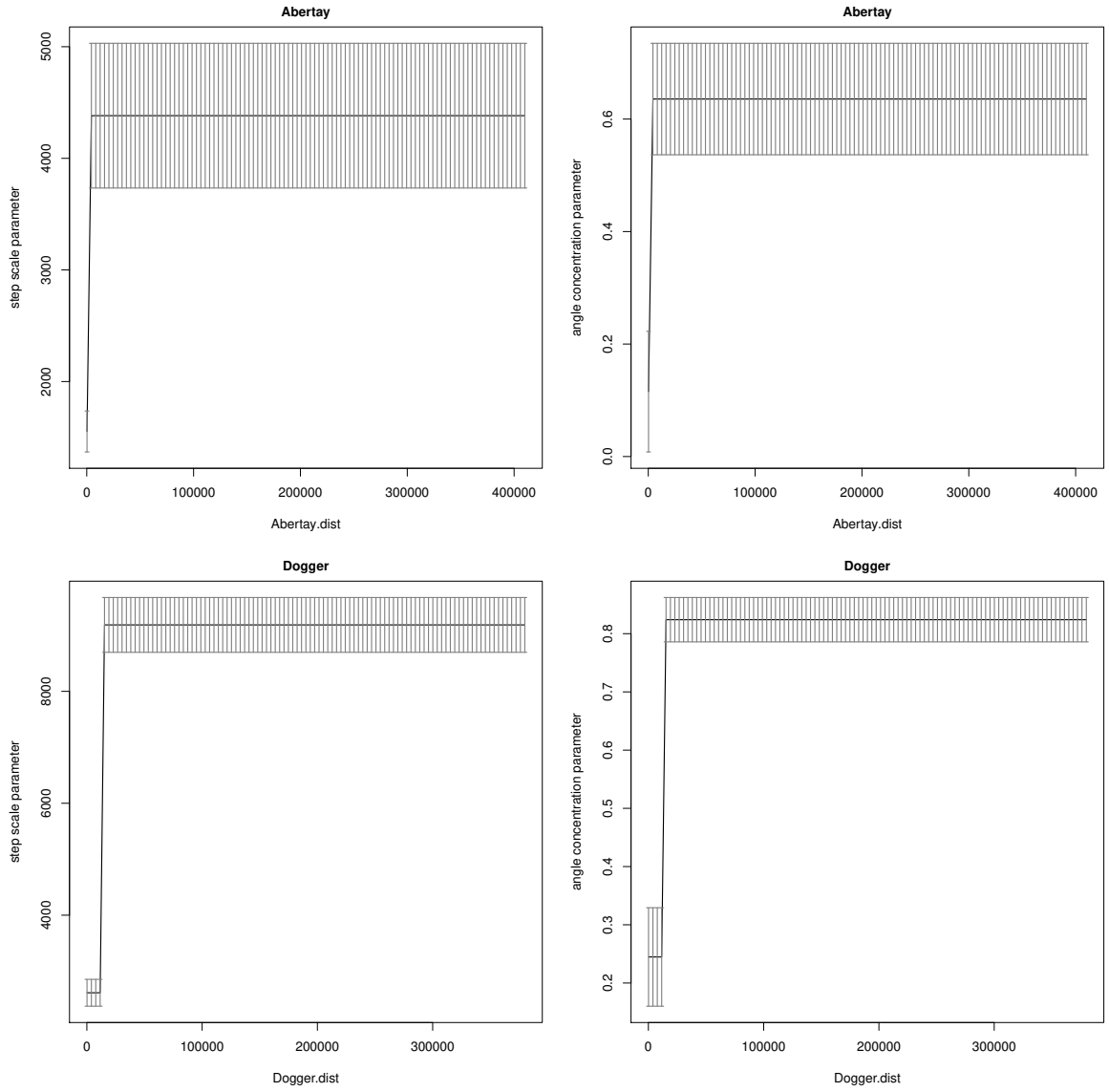

Figure 5. *Selected results from the grey seal example. Panels include estimates and 95% confidence intervals for the "Abertay" haul-out state step length scale parameter as a function of distance in meters ('Abertay.dist'; top-left panel), "Abertay" haul-out state turning angle concentration parameter as a function of distance (top-right panel), "Dogger Bank" foraging state step length scale parameter as a function of distance ('Dogger.dist'; bottom-left panel), and the "Dogger Bank" foraging state turning angle concentration parameter as a function of distance (bottom-right panel).*

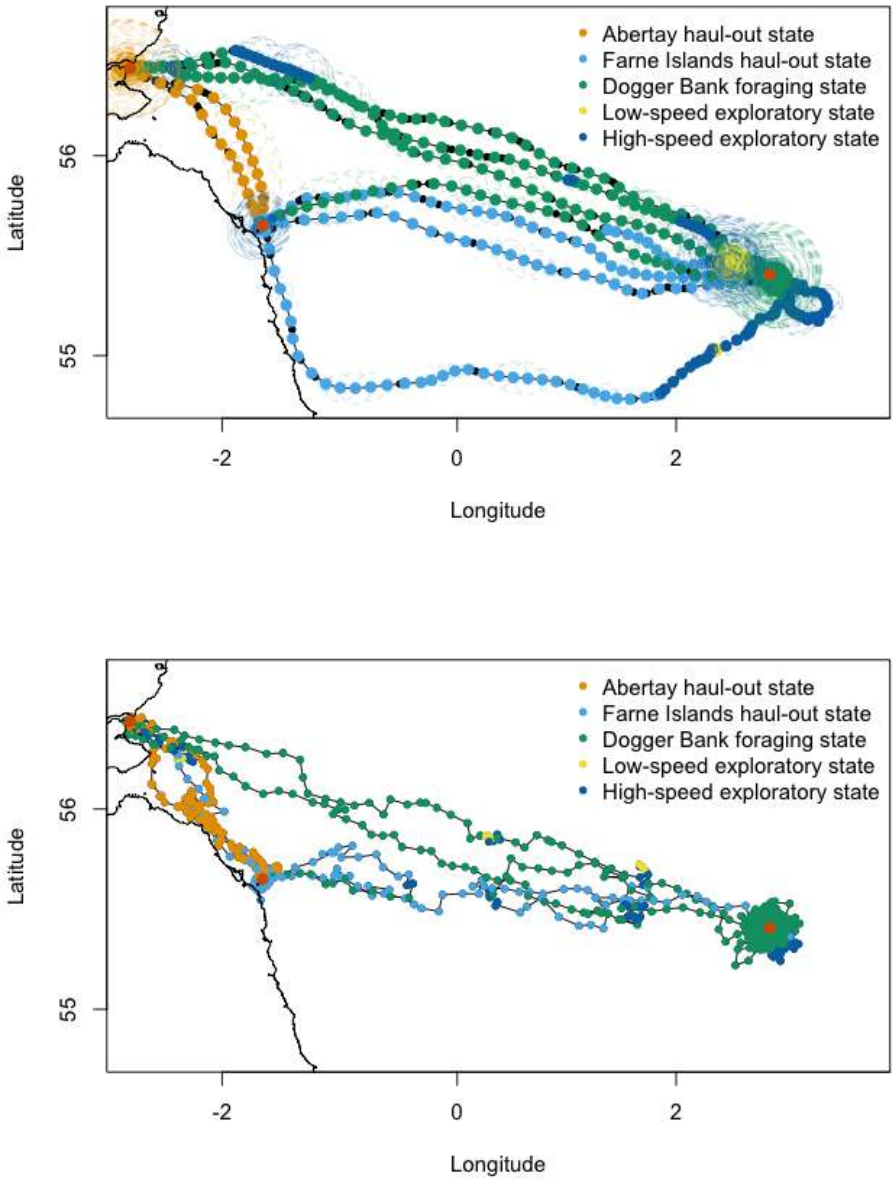

Figure 6. *Fitted and simulated tracks from the grey seal example. This seal tended to move in a clockwise fashion between two haul-out locations ("Abertay" and "Farne Islands") and a foraging area ("Dogger Bank") in the North Sea. Top panel plots the pooled track, 95% error ellipse confidence bands, and state estimates based on the 5-state HMM fitted to multiple imputations of the position process. Red points indicate the locations of the three activity centers. Black points indicate the (temporally-irregular) observed locations. Bottom panel presents the locations and states for a track simulated from the fitted model using the 'simData' function.*

(at most) one foraging trip, starting from Kerguelen Island. We want to fit the model described by Michelot *et al.* (2017), with the four following states:

1. outbound trip from the colony to the ice;
2. searching;
3. foraging;
4. inbound trip from the ice to the colony.

The data set has three columns: "ID" (track ID), "x" (longitude), and "y" (latitude):

```
head(tracks)
```

```
##   ID        x         y
## 1  1 70.60946 -49.60737
## 2  1 70.82908 -50.08287
## 3  1 70.90029 -50.32835
## 4  1 70.85766 -50.54746
## 5  1 70.63792 -50.90529
## 6  1 70.48480 -50.99666
```

From the locations, we use `prepData` to derive the step lengths and turning angles, as well as the distance and bearing (relative to previous movement direction as in Eq. 3) to the Kerguelen Island colony (with coordinates 70° longitude and -49° latitude):

```
center <- matrix(c(70,-49),nrow=1,dimnames=list("colony"))
data <- prepData(data=tracks, type="LL", centers=center)
```

Note that distances are in kilometers and angles are based on initial bearings (using geosphere::bearing; Hijmans 2016a) when calculated from longitude and latitude coordinates.

3.5.1 Model 1: no covariates

We start by fitting a covariate-free 4-state correlated random walk model, which we will use to extract starting parameter values for more complex models. We use the argument `fixPar` to fix some transition probabilities to zero, following Michelot *et al.* (2017). We set to `NA` the columns of unconstrained transition probabilities, and we fix

the intercept of the other columns to a large negative number (here -100) to set the corresponding transition probabilities to be virtually zero (i.e. impossible transition). As in Michelot *et al.* (2017), we set transition probabilities from outbound to forage, outbound to inbound, search to outbound, forage to outbound, forage to inbound, inbound to outbound, inbound to search, and inbound to forage to be effectively zero.

```
stateNames <- c("outbound","search","forage","inbound")

# initial parameters
stepPar0 <- c(25,5,1,25,10,5,3,10)
anglePar0 <- c(15,5,2,15)

# constrain transition probabilities
fixbeta <- matrix(c(NA,-100,-100,-100,NA,NA,-100,NA,-100,-100,-100,-100),
                  nrow=1)

m1 <- fitHMM(data=data, nbStates=4, dist=list(step="gamma",angle="vm"),
             Par0=list(step=stepPar0, angle=anglePar0),
             fixPar=list(beta=fixbeta), stateNames = stateNames)
```

3.5.2 Model 2

This model mimics the formulation of Michelot *et al.* (2017). We model the effect of the distance to colony on the transition probability from outbound to search, and of the time since departure on the transition probability from search to inbound.

```
# time spent since left colony
time <- NULL
for(id in unique(data$ID)) {
    nbSubObs <- length(which(data$ID==id))

    # approximately in months for interval = 9.6h
    time <- c(time, (1:nbSubObs)/75)
}

data$time <- time

# compute time since departure and include in formula below
formula <- ~ colony.dist + time
```

As before, we constrain the transition probability matrix to prevent some of the transitions (e.g. from forage to inbound, etc.). We define a 3×12 matrix for the beta parameters, in which each column corresponds to a transition ($1 \to 2, 1 \to 3, 1 \to 4, 2 \to 1, \ldots$), and each row corresponds to a covariate (intercept, distance to center, time since departure). We set to `NA` the columns of unconstrained transition probabilities, and we again fix the intercept of the other columns to a large negative number (here -100) to set the corresponding transition probabilities to be virtually zero (i.e. impossible transition).

```
fixbeta <- matrix(c(NA,-100,-100,-100,NA,NA,-100,NA,-100,-100,-100,-100,
                    NA,   0,   0,   0, 0, 0,   0, 0,   0,   0,   0,   0,
                     0,   0,   0,   0, 0,NA,   0, 0,   0,   0,   0,   0),
                  nrow=3,byrow=TRUE)
```

Biased random walks are used to model the movement in states 1 and 4, with repulsion away from the colony in the outbound trip, and attraction towards the colony in the inbound trip. For that purpose, we include 'colony.angle' as a covariate on the angle mean of the von Mises distributions for turning angles in states 1 and 4.

```
angleFormula <- ~ state1(colony.angle) + state4(colony.angle)
```

To specify the direction of the bias (away from or towards the colony), we fix the parameters linking the mean turning angle to the direction of the colony. In state 1, we fix the coefficient to a large negative number (e.g. -100) so that the mean direction is away from the colony. In state 4, we fix it to a large positive number (e.g. 100) so that the mean direction is towards the colony. Fixing these parameters to large positive (or negative) values effectively removes the correlated random walk component of Eq. 2 for states 1 and 4. The four other parameters correspond to the angle concentrations and should be estimated (NAs in `fixPar`).

```
fixPar <- list(angle=c(-100,100,NA,NA,NA,NA),beta=fixbeta)
```

Because no covariates are specified for the mean angle of state 2 (searching) and state 3 (foraging), these states are reduced to correlated random walks with a mean turning angle of zero (i.e. $\text{atan2}(0, 1) = 0$ as in Eq. 2).

We can now fit the second model with starting parameter values extracted from the simpler model using `getPar0`. In `fitHMM`, we use the arguments `estAngleMean` and `circularAngleMean` to indicate that the angle mean is to be estimated using circular-circular regression.

```
Par0 <- getPar0(model=m1, nbStates=4,
                DM=list(angle=list(mean=angleFormula, concentration=~1)),
                estAngleMean=list(angle=TRUE),
                circularAngleMean=list(angle=TRUE), formula=formula)

m2 <- fitHMM(data=data, nbStates=4, dist=list(step="gamma",angle="vm"),
             Par0=list(step=Par0$Par$step, angle=Par0$Par$angle),
             beta0=Par0$beta, fixPar=fixPar, formula=formula,
             DM=list(angle=list(mean=angleFormula, concentration=~1)),
             estAngleMean=list(angle=TRUE), circularAngleMean=list(angle=TRUE),
             stateNames = stateNames)
```

Instead of relying entirely on `fixPar` for parameter constraints, an equivalent model for the transition probabilities could be specified using the special function `betaCol` in `formula`:

```
formula <- ~ betaCol1(colony.dist) + betaCol6(time)

fixbeta <- matrix(c(NA,-100,-100,-100,NA,NA,-100,NA,-100,-100,-100,-100,
                    rep(NA,12),
                    rep(NA,12)),
                  nrow=3,byrow=TRUE)

fixPar <- list(angle=c(-100,100,NA,NA,NA,NA),beta=fixbeta)
```

Here `betaCol1(colony.dist)` specifies an effect of distance to colony only on the transition from state 1 to state 2 (which corresponds to the first column of the beta matrix) and `betaCol6(time)` specifies an effect of time since departure only on the transition from state 2 to state 4 (which corresponds to the sixth column of the beta matrix). When the special function `betaCol` is used, then `fitHMM` automatically fixes the appropriate elements in the second ('colony.dist') and third ('time') rows of the beta matrix to zero (without the user needing to do so manually using `fixPar`). However, note that the first row (corresponding to the intercept terms) must still be manually fixed to achieve the desired constraints on the transition probability matrix.

3.5.3 Model 3

In addition to the covariates included in model 2, we add the effect of distance to colony on the step length and turning angle concentration parameters for states 1 and 4. We specify the following formulas:

```
distFormula <- ~ state1(colony.dist) + state4(colony.dist)
stepDM <- list(mean=distFormula, sd=distFormula)
angleDM <- list(mean=angleFormula, concentration=distFormula)
```

The initial parameters are extracted from model 2, again using the function `getPar0`. Instead of fixing the mean direction of movement like in model 2, we estimate it here as a trade-off between short-term directional persistence and bias toward (or away) from the colony (i.e. a biased correlated random walk as in Eq. 2).

```
# remove fixed angle parameters
fixPar <- list(beta=fixbeta)

# get starting parameters from m2
Par0 <- getPar0(model=m2, nbStates=4,
                DM = list(step=stepDM, angle=angleDM),
                estAngleMean=list(angle=TRUE),
                circularAngleMean=list(angle=TRUE),
                formula=formula)

# the bias is estimated rather than fixed
Par0$Par$angle[c("mean_1:(colony.angle)","mean_4:(colony.angle)")] <- 0

m3 <- fitHMM(data=data, nbStates=4, dist=list(step="gamma",angle="vm"),
             Par0=list(step=Par0$Par$step, angle=Par0$Par$angle),
             beta0=Par0$beta, fixPar=fixPar, formula=formula,
             DM = list(step=stepDM, angle=angleDM),
             estAngleMean=list(angle=TRUE),
             circularAngleMean=list(angle=TRUE),
             stateNames = stateNames)
```

The three fitted model can be compared with `AIC(m1,m2,m3)`, which overwhelmingly supports model 3. The most likely state sequence is obtained with `viterbi(m3)`. Figure 7 shows a map of the state-decoded track. The estimated circular-circular regression coefficients for the angle means of state 1 (outbound) and state 4 (inbound)

were -0.66 (95% CI: $-0.82 - -0.5$) and 0.4 (95% CI: $0.32 - 0.49$), respectively, thus indicating biased correlated random walks with repulsion away from the colony during outbound movements and attraction towards the colony during inbound movements. The estimated regression coefficients for the step length mean and turning angle concentration parameters for states 1 and 4 suggest that step lengths decreased and turning angles became more concentrated at the mean angle as distance to colony increased (Figure 8)

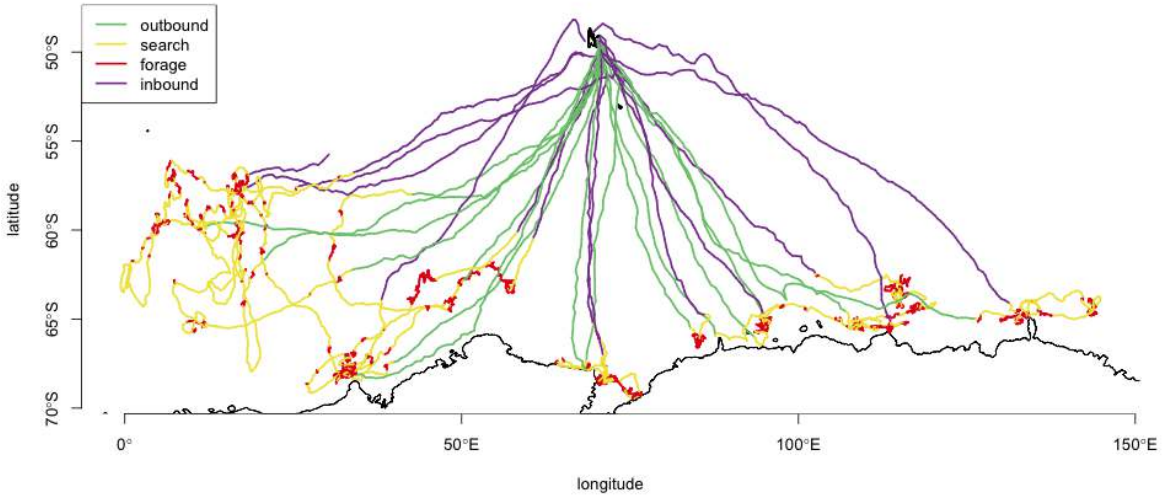

Figure 7. *The 15 elephant seal tracks, colored by the most likely state sequence.*

3.6 Group dynamic animal movement

Here we demonstrate how `momentuHMM` can be used to simulate and fit the group dynamic animal movement model of Langrock *et al.* (2014). In group dynamic models, groups (e.g., herds, packs, schools) are allowed to influence the movement of social individuals. One way to accomplish this is to model individual movements as being attracted to a group "centroid". Depending on the system, the centroid could simply be the location of the group center (e.g., the mathematical centroid of the group) or group leader (e.g., alpha wolf) at times $t = 1, \ldots, T$. In this sense, the centroid can be considered a dynamic activity center that changes position over time, and these models are not necessarily limited to groups. For example, the centroid could instead

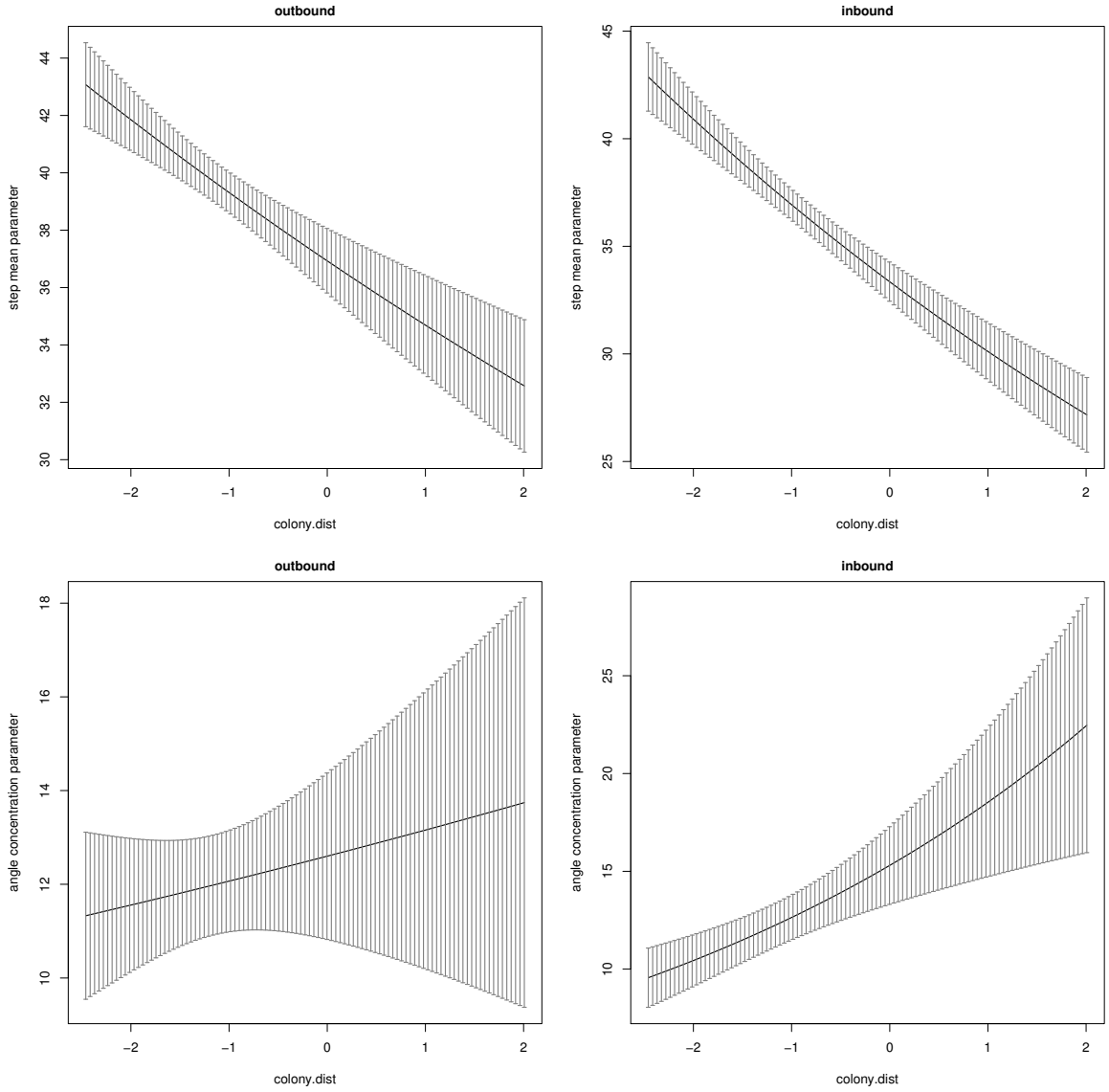

Figure 8. *Selected results from the elephant seal example. Panels include estimates and 95% confidence intervals for the "outbound" mean step length parameter (top-left panel), "inbound" mean step length parameter (top-right panel), "outbound" turning angle concentration parameter (bottom-left panel), and "inbound" turning angle concentration parameter (bottom-right panel) as a function of distance to colony ('colony.dist'). Distance to colony has been standardized based on a mean of 2539 km (SD = 1021.3).*

refer to predators, competitors, or human activity (in which case the centroid might be repulsive rather than attractive!).

Dynamic activity centers can be simulated in `simData` using the `centroids` argument. Following simulation scenario A of Langrock *et al.* (2014), we first simulate a group centroid as a single-state (i.e. $N = 1$) biased correlated random walk relative to the origin:

```
dist <- list(step="gamma", angle="vm")
nbObs <- 250

Parc <- list(step = c(15,10),
             angle = c(0.15,log(1)))

DMc <- list(angle=list(mean = ~center1.angle,
                       concentration=~1))

centroidData <- simData(nbStates=1, dist=dist, Par=Parc, DM=DMc,
                        circularAngleMean = list(angle = TRUE),
                        centers = matrix(0,1,2),
                        obsPerAnimal = nbObs)
```

Now we can use the simulated centroid track (Fig. 9) as a dynamic activity center and simulate the movement of a group of 20 individuals as a 2-state mixture of a biased random walk (relative to the centroid) and a correlated random walk (independent of the centroid):

```
nbAnimals <- 20
nbStates <- 2
stateNames <- c("group","solitary")

Par <- list(step = c(30,50,15,25),
            angle = c(1.e+7,log(2.5),log(5)))

beta <- matrix(c(-2.944439,-1.734601),1,nbStates)

DM <- list(angle=list(mean = ~state1(centroid.angle),
                      concentration = ~1))

# calculate stationary distribution
gamma <- diag(nbStates)
```

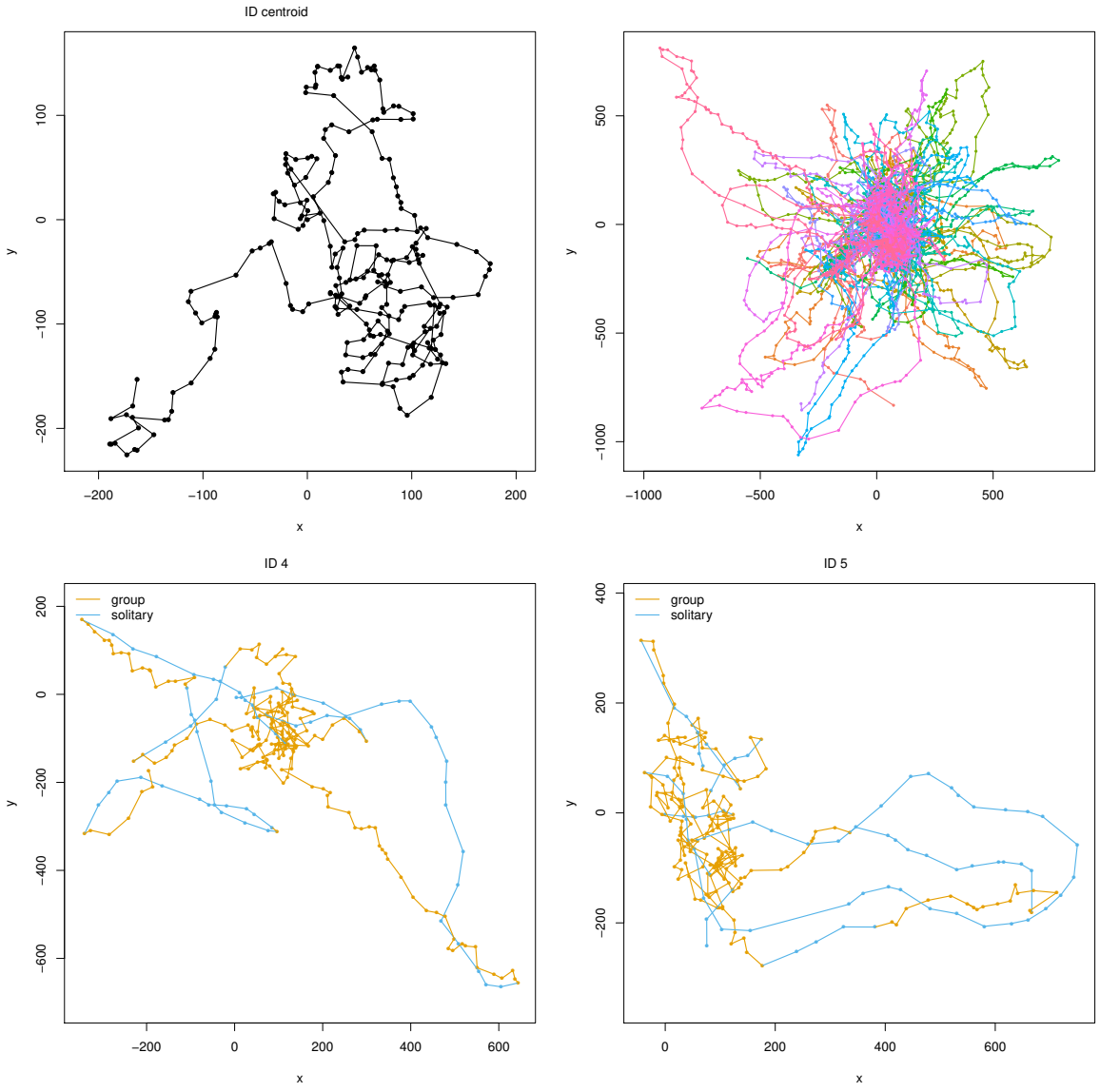

Figure 9. *Selected results from the group dynamic animal movement example. Panels include the simulated centroid path (top-left panel), the simulated paths of 20 individuals where state 1 ("group") includes biased movements towards the centroid and state 2 ("solitary") is a correlated random walk independent of the group centroid (top-right panel), and two fitted tracks that are colored by the most likely state sequence (bottom panels).*

```
gamma[!gamma] <- exp(beta)
gamma <- t(gamma)
gamma <- gamma/apply(gamma,1,sum)
delta <- solve(diag(nbStates) - t(gamma) + 1, rep(1, nbStates))

# draw random initial locations for each individual
initialPositions <- vector("list")
for (i in 1:nbAnimals) {
  initialPositions[[i]] <- runif(2, -10, 10)
}

# create centroid data frame
cD <- data.frame(x = centroidData$x, y = centroidData$y)

groupData <- simData(nbAnimals=nbAnimals, nbStates=nbStates, dist=dist,
                     Par = Par, beta = beta, delta = delta, DM = DM,
                     circularAngleMean = list(angle = TRUE),
                     centroids = list(centroid = cD),
                     obsPerAnimal = nbObs,
                     initialPosition = initialPositions,
                     states = TRUE, stateNames = stateNames)
```

Here state 1 ("group") has biased movements toward the centroid and state 2 ("solitary") is simply a correlated random walk independent of the group centroid (Fig. 9). Note that despite this being a 2-state HMM, the working scale parameters for turning angle (`Par$angle`) only includes 3 parameters (1 for the angle mean and 2 for the concentration parameters). This is because under the circular-circular regression model, no working parameter is specified[2] for the reference turning angle of zero (i.e., the component for short-term directional persistence; see Eq. 2) and no angular covariates were specified in the model for state 2 ("solitary"). Thus the first parameter corresponds to the working scale parameter of the `centroid.angle` covariate for state 1 ("group"), while the second and third parameters are the working scale parameters for the concentration parameters for states 1 and 2, respectively. In this case, the angle mean parameter for state 1 was set at a large value (1.e+7) so as to enforce a biased random walk (instead of a biased correlated random walk). In other words, the large value for the biased random walk component "swamps" the correlated random walk component in Eq. 2.

[2]More accurately, the working parameter for the reference angle is automatically fixed to $\beta_0 = 1$.

Finally, we can fit the group dynamic model using `fitHMM`:

```
Par0 <- list(step = c(30,50,15,25),
             angle = c(0,log(2.5),log(5)))

groupFit <- fitHMM(groupData, nbStates=nbStates, dist=dist, Par=Par0,
                   DM = DM, stationary = TRUE,
                   estAngleMean = list(angle = TRUE),
                   circularAngleMean = list(angle = TRUE),
                   stateNames = stateNames)
```

3.7 Harbour seals

Here we demonstrate how more complicated parameter constraints can be implemented using the `userBounds` and `workBounds` arguments in `fitHMM` and `MIfitHMM`. This example is based on the harbour seal analysis of McClintock *et al.* (2013). Using individual-level random effects on probability distribution parameters, McClintock *et al.* (2013) performed a Bayesian analysis of population-level activity budgets for 3-states ("resting", "foraging", and "transit"). While `momentuHMM` cannot be used to replicate this analysis exactly, we can perform a similar analysis in the absence of individual-level random effects. Here we will focus on several specific parameter constraints, but the full example code can be found in the "vignettes" source directory.

The harbour seal data consist of 17 individuals (10 male, 7 female) and, as in the northern fur example in section 3.2, both location and dive activity data. The location data were obtained at temporally-irregular intervals, while the dive activity data were obtained at regular 2-hour time steps. We therefore first used `crawlWrap` to fit and predict locations for all 17 tracks at 2-hour time steps and then used `crawlMerge` to merge the predicted locations with the dive activity data. We then fitted several different models assuming: 1) no individual- or sex-level effects on all parameters (i.e., the "null" model); 2) sex-level effects on all parameters; and 3) individual-level effects on all parameters. Based on `fitHMM` fits for the best predicted tracks, AIC overwhelmingly supported the model including individual-level effects on all parameters, but for simplicity we will use the model including no individual- or sex-level effects to demonstrate how the constraints of McClintock *et al.* (2013) can be implemented in `momentuHMM`. In addition to the lack of individual-level random effects in our example, we also depart

from McClintock *et al.* (2013) in our use of zero-inflation parameters to account for steps of length zero (i.e., $l_t = 0$) and time steps with no dive activity (i.e., $\omega_t = 0$). Unlike McClintock *et al.* (2013), note that our model 3 also includes individual-level fixed effects on the state transition probabilities.

McClintock *et al.* (2013) fit their 3-state model using more complicated constraints on the probability distribution parameters than any of our previous vignette examples. In addition to relational constraints among the "resting", "foraging", and "transit" states similar to those used in the northern fur seal example (section 3.2), these constraints included upper bounds for the shape and scale parameters of the Weibull distribution for step length, a minimum value for the "transit" concentration parameter of the wrapped Cauchy distribution for turning angle, and bounds on the shape parameters of the beta distribution for dive activity specifically chosen to prevent any "bathtub" shaped distributions for the proportion of each time step spent diving below 1.5m.

Before we demonstrate how to implement these constraints, we should first provide more detail on exactly how the `DM`, `userBounds`, and `workBounds` arguments work together in `fitHMM` (and `MIfitHMM`). While the `DM` argument should now be familiar, we have thus far spent little time discussing the latter two arguments. The `userBounds` argument specifies the lower and upper bounds for the natural scale parameters as a 2-column matrix. By default all working scale parameters ($\boldsymbol{\beta}_\theta$) are bounded on the real line, but the `workBounds` argument can be used to specify the lower and upper bounds for the working scale parameters as a 2-column matrix. Specifically, `momentuHMM` calculates natural scale parameters with finite bounds as

$$\boldsymbol{\theta} = (\mathbf{U}_\theta - \mathbf{L}_\theta)\, g^{-1}\left(\mathbf{X}_\theta \boldsymbol{\beta}^*_\theta\right) + \mathbf{L}_\theta,$$

where $\mathbf{L}_\theta$ is the lower bound on the natural scale and $\mathbf{U}_\theta$ is the upper bound on the natural scale. Note that $\boldsymbol{\beta}^*_\theta = \boldsymbol{\beta}_\theta$ under the default values for `workBounds`. For natural scale parameters with finite lower bounds and infinite upper bounds, we have

$$\boldsymbol{\theta} = g^{-1}\left(\mathbf{X}_\theta \boldsymbol{\beta}^*_\theta\right) + \mathbf{L}_\theta.$$

When `workBounds` is specified, then additional link functions are used on the working scale parameters. For example, for working scale parameters with finite bounds, we

have

$$\boldsymbol{\beta}_\theta^* \;=\; (\mathbf{U}_{\beta_\theta} - \mathbf{L}_{\beta_\theta})\,\text{logit}^{-1}\left(\boldsymbol{\beta}_\theta\right) + \mathbf{L}_{\beta_\theta},$$

where $\mathbf{L}_{\beta_\theta}$ is the lower bound on the working scale and $\mathbf{U}_{\beta_\theta}$ is the upper bound on the working scale. When constraining working parameters with finite lower bounds (e.g. zero) and infinite upper bounds, we have

$$\boldsymbol{\beta}_\theta^* = \exp\left(\boldsymbol{\beta}_\theta\right) + \mathbf{L}_{\beta_\theta}.$$

When constraining working parameters with infinite lower bounds and finite upper bounds (e.g. zero), we have

$$\boldsymbol{\beta}_\theta^* = -\left(\exp\left(-\boldsymbol{\beta}_\theta\right) - \mathbf{U}_{\beta_\theta}\right).$$

Although optimization within `fitHMM` and `MIfitHMM` is always performed on $\boldsymbol{\beta}_\theta$, note that $\boldsymbol{\beta}_\theta^*$ (and a delta method approximation for the variance of this transformation) is returned by `CIbeta`, `MIpool`, and `print` function calls.

For the Weibull distribution parameters for step length, McClintock *et al.* (2013) constrained the shape parameter $a_s < 5$ (to prevent too "peaked" distributions) and the scale parameter less than the maximum distance a harbour seal could travel in 2 hours at 2 m/s (i.e., $b_s < 14400$m) for $s \in \{1 = \text{"resting"}, 2 = \text{"foraging"}, 3 = \text{"transit"}\}$. They also constrained $b_3 \geq b_2 \geq b_1$. This is easily accomplished in `momentuHMM` using the `DM`, `userBounds`, and `workBounds` arguments:

```
nbStates <- 3
stateNames <- c("resting", "foraging", "transit")
dist <- list(step = "weibull", angle = "wrpcauchy", dive = "beta")
stepDM<-matrix(c(1,0,0,0,0,0,0,0,0,
                 0,1,0,0,0,0,0,0,0,
                 0,0,1,0,0,0,0,0,0,
                 0,0,0,1,0,0,0,0,0,
                 0,0,0,1,1,0,0,0,0,
                 0,0,0,1,1,1,0,0,0,
                 0,0,0,0,0,0,1,0,0,
                 0,0,0,0,0,0,0,1,0,
                 0,0,0,0,0,0,0,0,1),nrow=3*nbStates,byrow=TRUE,
       dimnames=list(c(paste0("shape_",1:nbStates),
```

```
                        paste0("scale_",1:nbStates),
                        paste0("zeromass_",1:nbStates)),
                      c(paste0("shape_",1:nbStates,":(Intercept)"),
                        "scale:(Intercept)","scale_2","scale_3",
                        paste0("zeromass_",1:nbStates,":(Intercept)"))))
stepworkBounds<-matrix(c(rep(-Inf,4),0,0,rep(-Inf,3),
                   rep(Inf,ncol(stepDM))),ncol(stepDM),2,
                   dimnames=list(colnames(stepDM),c("lower","upper")))
stepBounds<-matrix(c(0,5,
                     0,5,
                     0,5,
                     0,14400,
                     0,14400,
                     0,14400,
                     0,1,
                     0,1,
                     0,1),nrow=3*nbStates,byrow=TRUE,
                    dimnames=list(rownames(stepDM),c("lower","upper")))
```

When included in `workBounds` and `userBounds` for the step length data stream, 'stepworkBounds' and 'stepBounds' above constrain the parameters $\beta^*_{l,5} > 0$ and $\beta^*_{l,6} > 0$ such that $14400 \geq b_3 \geq b_2 \geq b_1 \geq 0$:

$$\begin{aligned} b_1 &= (14400-0)\,\text{logit}^{-1}\left(\beta_{l,4}\right)+0 \\ b_2 &= (14400-0)\,\text{logit}^{-1}\left(\beta_{l,4}+\exp(\beta_{l,5})+0\right)+0 \\ b_3 &= (14400-0)\,\text{logit}^{-1}\left(\beta_{l,4}+\exp(\beta_{l,5})+0+\exp(\beta_{l,6})+0\right)+0. \end{aligned}$$

In order to force the "transit" state to have strong directional persistence, McClintock *et al.* (2013) constrained the concentration parameter $\rho_3 > 0.75$. In the absence of relational constraints, `userBounds` could be used to constrain $\rho_3 > 0.75$. However, because we also wish to constrain $\rho_2 \leq \rho_3$, we must make use of the `workBounds` argument. We can constrain $\rho_3 \geq 0.75$, $\rho_2 \leq \rho_3$, and $\rho_s \leq 0.95$ ($s \in \{1,2,3\}$) using the following combination of `DM`, `userBounds`, and `workBounds` arguments:

```
angleDM <- matrix(c(1,0,0,
                    0,1,1,
                    0,1,0),nrow=nbStates,byrow=TRUE,
                  dimnames=list(paste0("concentration_",1:nbStates),
                                c("concentration_1:(Intercept)",
```

```
                                    "concentration_23:(Intercept)",
                                    "concentration_2")))
angleBounds <- matrix(c(0,0.95,
                        0,0.95,
                        0,0.95),nrow=nbStates,byrow=TRUE,
                      dimnames=list(rownames(angleDM),c("lower","upper")))
transitcons <- boot::logit((0.75 - angleBounds[3,1])
                           /(angleBounds[3,2] - angleBounds[3,1]))
angleworkBounds <- matrix(c(-Inf,transitcons,-Inf,
                            rep(Inf,2),0),ncol(angleDM),2,
                          dimnames=list(colnames(angleDM),c("lower","upper")))
```

When 'angleworkBounds' and 'angleBounds' are respectively included in `workBounds` and `userBounds` for the turning angle data stream, this yields

$$\begin{aligned}
\rho_1 &= (0.95-0)\,\text{logit}^{-1}(\beta_{\phi,1}) + 0 \\
\rho_2 &= (0.95-0)\,\text{logit}^{-1}(\exp(\beta_{\phi,2}) + 1.32 - (\exp(-\beta_{\phi,3}) - 0)) + 0 \\
\rho_3 &= (0.95-0)\,\text{logit}^{-1}(\exp(\beta_{\phi,2}) + 1.32) + 0.
\end{aligned}$$

Here we constrained $\rho_s \leq 0.95$ to avoid numerical convergence issues that can arise with sparse data sets as $\rho_s \to 1$. Also note that when using `workBounds` to enforce a specific constraint on the natural scale, `userBounds` should not also include the corresponding natural parameter constraint(s). For example, because we are constraining $\rho_3 \geq 0.75$ with the `workBounds` argument, we did not also include a 0.75 lower bound for ρ_3 in 'angleBounds' above.

For the beta distribution of the dive activity data, McClintock *et al.* (2013) constrained the shape1 (υ_s) and shape2 (δ_s) parameters as follows:

$$\begin{aligned}
1 \leq \upsilon_1 \leq \delta_1 \leq 10 \\
1 \leq \delta_2 \leq \upsilon_2 \leq 10
\end{aligned}$$

where $\upsilon_2 = \upsilon_3$ and $\delta_2 = \delta_3$. These constraints can be imposed using the following combination of `DM`, `userBounds`, and `workBounds` arguments:

```
omegaDM <- matrix(c(1,0,0,0,0,0,
                    0,0,1,1,0,0,
                    0,0,1,1,0,0,
```

```
                    1,1,0,0,0,0,
                    0,0,1,0,0,0,
                    0,0,1,0,0,0,
                    0,0,0,0,1,0,
                    0,0,0,0,0,1,
                    0,0,0,0,0,1),nrow=nbStates*3,byrow=TRUE,
                dimnames=list(c(paste0("shape1_",1:nbStates),
                                paste0("shape2_",1:nbStates),
                                paste0("zeromass_",1:nbStates)),
                              c("shape_1:(Intercept)","shape2_1",
                                "shape_2:(Intercept)","shape1_2",
                                "zeromass_1:(Intercept)",
                                "zeromass_23:(Intercept)")))
omegaworkBounds <- matrix(c(-Inf,0,-Inf,0,-Inf,-Inf,
                            rep(Inf,ncol(omegaDM))),ncol(omegaDM),2,
                          dimnames=list(colnames(omegaDM),c("lower","upper")))
omegaBounds <- matrix(c(1,10,
                        1,10,
                        1,10,
                        1,10,
                        1,10,
                        1,10,
                        0,1,
                        0,1,
                        0,1),nrow=nbStates*3,byrow=TRUE,
                      dimnames=list(rownames(omegaDM),c("lower","upper")))
```

Lastly, we wish to impose some biologically-meaningful constraints on the zero-inflation parameters using the `fixPar` argument. Because we would never expect a harbour seal in the "transit" state to exhibit a step length of zero, it makes sense to constrain the zero-mass step length parameter for the "transit" state to (effectively) zero. Similarly, we would not expect a harbour seal in the "foraging" or "transit" states to exhibit no dive activity, and it therefore also makes sense to constrain the zero-mass dive activity parameters for these states to zero. We can accomplish this using the following `fixPar` argument:

```
fixPar <- list(step=c(rep(NA,nbStates*2),NA,NA,boot::logit(1.e-100)),
               omega=c(rep(NA,4),NA,boot::logit(1.e-100)))
```

Putting it all together, we can fit our constrained model assuming no individual- or sex-level effects using `MIfitHMM`:

```
DM <- list(step = stepDM, angle = angleDM, omega = omegaDM)
userBounds <- list(step = stepBounds,
                   angle = angleBounds,
                   omega = omegaBounds)
workBounds <- list(step = stepworkBounds,
                   angle = angleworkBounds,
                   omega = omegaworkBounds)
hsFits <- MIfitHMM(crwOut, nSims = 30,
                   nbStates = nbStates, dist = dist, Par0 = Par0,
                   DM = DM, workBounds = workBounds,
                   userBounds = userBounds, workBounds = workBounds,
                   fixPar = fixPar, stateNames = stateNames)
```

As was mentioned earlier, we found overwhelming AIC support for the individual-level effects model relative to the sex-level effects models and the null model above. While our best supported `momentuHMM` model included individual-level fixed effects, the estimated tracks (Fig. 10) and inferences about population-level activity budgets were similar to the individual-level random effects model of McClintock *et al.* (2013). Estimated activity budgets for the males were 0.36 (95% CI: $0.35 - 0.37$) for "resting", 0.53 (95% CI: $0.52 - 0.54$) for "foraging", and 0.11 (95% CI: $0.1 - 0.12$) for "transit". Activity budgets for females were 0.3 (95% CI: $0.29 - 0.3$) for "resting", 0.61 (95% CI: $0.6 - 0.62$) for "foraging", and 0.09 (95% CI: $0.08 - 0.1$) for "transit". We found considerable individual variation in the state transition probabilities (Fig. 11), and when comparing the estimated activity budgets of our analysis with those of McClintock *et al.* (2013), we suspect the more noticeable differences between the time spent in the "resting" and "foraging" states for males is attributable to our having included individual-level effects on the state transition probabilities.

3.8 Northern fulmars

Using Bayesian analysis methods, Pirotta *et al.* (2018) fit a 6-state biased random walk model to northern fulmar (*Fulmarus glacialis*) tracks in northern Scotland, UK. These states included biased movements relative to a colony in Eynhallow (59.12° N, 3.1° W) and fishing vessels that frequently work in the area. Pirotta *et al.* (2018) framed their

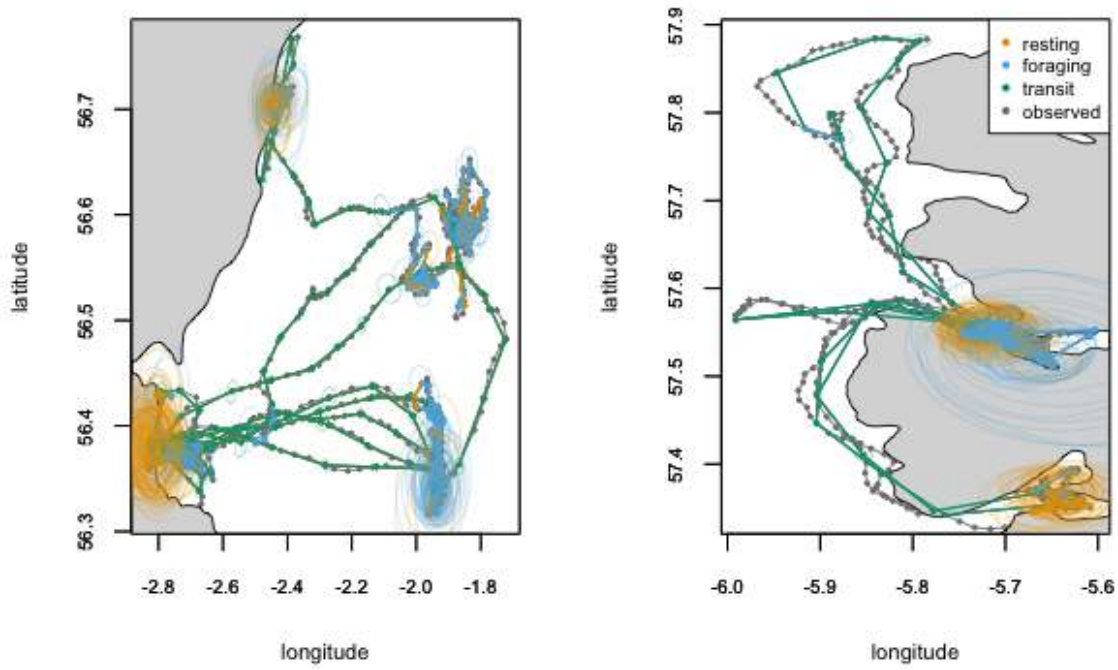

Figure 10. *Two harbour seal tracks, colored by the most likely state sequence.*

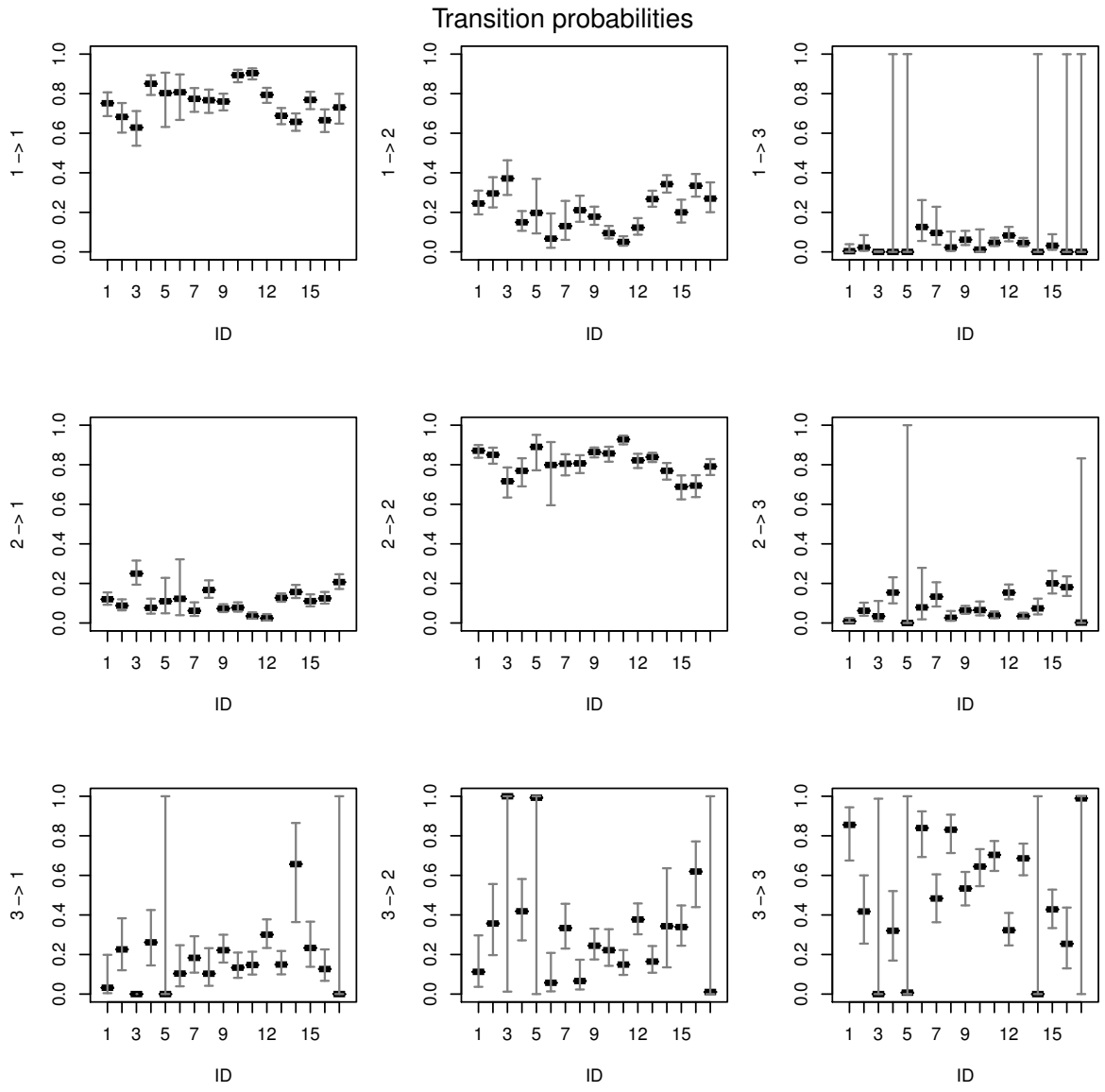

Figure 11. *Estimated individual-level state transition probabilities for the harbour seal example.*

model as having two latent state process. Under the first state process, the direction of movement could be biased away from the colony ("sea"), towards the nearest fishing vessel ("boat"), or towards the colony ("colony"). Under the second state process, the movement mode could either be fast and directionally-persistent ("transit") or area-restricted search ("ARS"). Thus the six states are "sea ARS", "sea transit", "boat ARS", "boat transit", "colony ARS" and "colony transit". Pirotta *et al.* (2018) also allowed the distance to the nearest fishing vessel and time since leaving the colony to affect state transitions to the "boat" and "colony" states. Here we demonstrate how a very similar (but not identical) model can be implemented in `momentuHMM`.

The data are provided in a Dryad repository, but these will require some additional formatting and preparation. We first load the raw data, create the required "ID" column based on individual trips, convert time stamps to class `POSIXct`, and project the northern fulmar ("Longitude", "Latitude") and nearest fishing vessel ("Boat_Longitude", "Boat_Latitude") locations using the `sp` package:

```
library(sp)

# load data provided by Pirotta et al
fulmarURL <- "https://datadryad.org/bitstream/handle/10255/dryad.174482/"
fileName <- "Fulmar_trackingData.csv?sequence=1"
raw_data <- read.csv(url(paste0(fulmarURL,fileName)),
                     stringsAsFactors = FALSE)

raw_data$ID <- raw_data$tripID
raw_data$Date <- as.POSIXct(raw_data$Date,tz="UTC",
                            format="%d/%m/%Y %H:%M")

# project data
oldProj <- CRS("+proj=longlat +datum=WGS84")
newProj <- CRS("+init=epsg:27700")
coordinates(raw_data) <- c("Longitude","Latitude")
proj4string(raw_data) <- oldProj
raw_data <- as.data.frame(spTransform(raw_data, newProj))

coordinates(raw_data) <- c("Boat_Longitude","Boat_Latitude")
proj4string(raw_data) <- oldProj
raw_data <- as.data.frame(spTransform(raw_data, newProj))
```

For movements away from the colony ("sea ARS" and "sea transit"), Pirotta *et al.*

(2018) included bias in the direction of the farthest location from the colony for a given trip. We can use the `centers` argument of `prepData` to identify these locations ("max_dist") and then calculate the expected angle for the "sea" states ("sea.angle") using `momentuHMM:::distAngle`:

```
# use prepData to calculate colony distance covariate ('sea.angle')
colony <- data.frame(x = -3.1, y = 59.12)
coordinates(colony) <- c("x", "y")
proj4string(colony) <- oldProj
colony <- as.matrix(as.data.frame(spTransform(colony, newProj)))
rownames(colony) <- "colony"
colony_dist <- prepData(raw_data, coordNames = c("Longitude","Latitude"),
                        centers = colony)

# calculate "sea" mean angle covariate
sea.angle <- NULL
for(id in unique(colony_dist$ID)) {
  idat <- subset(colony_dist,ID==id)
  nbSubObs <- length(which(colony_dist$ID==id))
  max_dist <- as.numeric(idat[which.max(idat$colony.dist),c("x","y")])
  max_angle <- momentuHMM:::distAngle(colony,colony,max_dist)[2]
  sea.angle <- c(sea.angle, rep(max_angle,nbSubObs))
}
raw_data$sea.angle <- sea.angle
```

Next we calculate the time since leaving colony covariate ("time"):

```
# calculate time since left colony covariate ('time')
time <- aInd <- NULL
for(id in unique(raw_data$ID)) {
  idInd <- which(raw_data$ID==id)
  aInd <- c(aInd,idInd[1])
  nbSubObs <- length(idInd)
  time <- c(time, (1:nbSubObs)/nbSubObs)
}
raw_data$time <- time
```

To complete our data preparation, we convert the nearest fishing vessel data to the `centriods` argument format and use `prepData` to calculate step lengths, turn angles, and our "sea", "boat", and "colony" covariates:

```
# get boat data into centroids argument format
boat_data <- list(boat=data.frame(Date = raw_data$Date,
                                  x = raw_data$Boat_Longitude,
                                  y = raw_data$Boat_Latitude))

# format and merge all data and covariates for analysis
fulmar_data <- prepData(raw_data, coordNames = c("Longitude","Latitude"),
                        centers = colony,
                        centroids = boat_data,
                        covNames = "time",
                        angleCovs = "sea.angle")

# momentuHMM doesn't like data streams and covariates to have same name,
# so create identical data column with different name
fulmar_data$d <- fulmar_data$boat.dist

# standarize boat.dist covariate
fulmar_data$boat.dist <- scale(fulmar_data$boat.dist)
```

Note that we use the `centers` argument for the colony (because its location is static) and the `centroids` argument for the nearest fishing vessel (because its location is dynamic).

Now that we've formatted the data, we're ready to specify the 6-state HMM. Using 10 min time steps, Pirotta *et al.* (2018) included three data streams in their model: step length ("step"), turn angle ("angle"), and distance to nearest boat ("d"). These were respectively modelled using Weibull, wrapped Cauchy, and log-normal distributions:

```
nbStates <- 6

stateNames <- c("seaARS", "seaTr",
                "boatARS", "boatTr",
                "colonyARS", "colonyTr")

dist <- list(step = "weibull",
             angle = "wrpcauchy",
             d = "lnorm")
```

Similar to the harbour seal example (section 3.7), Pirotta *et al.* (2018) used relational parameter constraints that can be specified in `momentuHMM` using pseudo-design matrices:

```
# specify data stream probability distribution parameter constraints
stepDM <- matrix(c(1,0,0,0,
                   0,1,0,0,
                   1,0,0,0,
                   0,1,0,0,
                   1,0,0,0,
                   0,1,0,0,
                   0,0,1,0,
                   0,0,1,1,
                   0,0,1,0,
                   0,0,1,1,
                   0,0,1,0,
                   0,0,1,1),2*nbStates,4,byrow=TRUE,
                 dimnames=list(c(paste0("shape_",1:nbStates),
                                 paste0("scale_",1:nbStates)),
                               c("shape:ARS","shape:Tr",
                                 "scale:(Intercept)","scale:Tr")))

# constrain scale parameters such that Tr > ARS
stepworkBounds <- matrix(c(-Inf,Inf,
                           -Inf,Inf,
                           -Inf,Inf,
                           0,Inf),ncol(stepDM),2,byrow=TRUE,
                         dimnames=list(colnames(stepDM),c("lower","upper")))

# include trip-level effects on angle mean concentration parameter
nbTrips <- length(unique(fulmar_data$ID))
angleDM <- matrix(c("sea.angle",0,0,0,0,rep(0,2*nbTrips),
                    "sea.angle",0,0,0,0,rep(0,2*nbTrips),
                    0,"boat.angle",0,0,0,rep(0,2*nbTrips),
                    0,"boat.angle",0,0,0,rep(0,2*nbTrips),
                    0,0,"colony.angle",0,0,rep(0,2*nbTrips),
                    0,0,"colony.angle",0,0,rep(0,2*nbTrips),
                    0,0,0,1,0,paste0("ID",1:nbTrips),rep(0,nbTrips),
                    0,0,0,1,1,paste0("ID",1:nbTrips),paste0("ID",1:nbTrips),
                    0,0,0,1,0,rep(0,2*nbTrips),
                    0,0,0,1,1,rep(0,2*nbTrips),
                    0,0,0,1,0,rep(0,2*nbTrips),
                    0,0,0,1,1,rep(0,2*nbTrips)),2*nbStates,3+2+2*nbTrips,byrow=TRUE,
                  dimnames=list(c(paste0("mean_",1:nbStates),
                                  paste0("concentration_",1:nbStates)),
                                c("mean:sea","mean:boat","mean:colony",
```

```
                                    "concentration:(Intercept)","concentration:Tr",
                                    paste0("concentration:ID",1:nbTrips,":(Intercept)"),
                                    paste0("concentration:ID",1:nbTrips,":Tr"))))

# constrain concentration parameters such that Tr > ARS
angleworkBounds <- matrix(c(-Inf,Inf,
                            -Inf,Inf,
                            -Inf,Inf,
                            -Inf,Inf,
                            0,Inf,
                            rep(c(-Inf,Inf),nbTrips),
                            rep(c(0,Inf),nbTrips)),ncol(angleDM),2,byrow=TRUE,
                          dimnames=list(colnames(angleDM),c("lower","upper")))

dDM <- matrix(c(1,1,0,0,
                1,1,0,0,
                1,0,0,0,
                1,0,0,0,
                1,1,0,0,
                1,1,0,0,
                0,0,1,1,
                0,0,1,1,
                0,0,1,0,
                0,0,1,0,
                0,0,1,1,
                0,0,1,1),2*nbStates,4,byrow=TRUE,
              dimnames=list(c(paste0("location_",1:nbStates),
                              paste0("scale_",1:nbStates)),
                            c("location:(Intercept)","location:noboat",
                              "scale:(Intercept)","scale:noboat")))

# constrain location and scale parameters such that sea and colony > boat
dworkBounds <- matrix(c(-Inf,Inf,
                        0,Inf,
                        -Inf,Inf,
                        0,Inf),ncol(dDM),2,byrow=TRUE,
                      dimnames=list(colnames(dDM),c("lower","upper")))

DM <- list(step = stepDM, angle = angleDM, d = dDM)

workBounds <- list(step = stepworkBounds,
                   angle = angleworkBounds,
```

```
                    d = dworkBounds)
```

To complete our model specification, we use the `toState` special function to model transitions to the "boat" and "colony" states as a function of distance to nearest fishing vessel ("boat.dist") and time since leaving colony ("time"), respectively. Following Pirotta *et al.* (2018), we will use the `knownStates` argument to fix the initial state to "sea transit". We will also use the `fixPar` argument to fix the initial state probabilities (because we are assuming these are known) and, as in section 3.6, constrain the model to a biased random walk by fixing the mean angle working scale parameters to a large positive value:

```
# state transition formula similar to Pirotta et al
formula <- ~ toState3(boat.dist) + toState4(boat.dist) +
             toState5(time) + toState6(time)

# specify knownStates
# Priotta et al assumed all animals start in state 2 ('seaTr')
knownStates <- rep(NA,nrow(fulmar_data))
knownStates[aInd] <- 2

# fix delta_2 = 1 because assuming initial state is known for each track
fixPar <- list(delta=c(100,rep(0,nbStates-2)))
fixPar$delta <- exp(c(0,fixPar$delta))/sum(exp(c(0,fixPar$delta)))
# Constrain model to BRW (instead of BCRW)
fixPar$angle <- c(rep(1.e+7, 3), rep(NA, 2+2*nbTrips))
```

Lastly, we used the `betaCons` argument to impose similar constraints as Pirotta *et al.* (2018) for the transition probability parameters. We accomplish this by using `betaCons` to set the transition probability working parameter intercept terms equal among the three "ARS" states and the three "transit" states. We also use `betaCons` to constrain the effects of 'boat.dist' and 'time' to be identical for each of the two movement modes ("ARS" and "transit") within each of the three biased movement states ("sea", "boat", and "colony"). `betaCons` must be a matrix of the same dimension as `beta0` and be composed of integers, where each beta working parameter is sequentially indexed in a column-wise fashion. Equality constraints can then be incorporated by having parameters share the same index. In this example we have:

```
betaCons

##             1 -> 2 1 -> 3 1 -> 4 1 -> 5 1 -> 6 2 -> 1 2 -> 3 2 -> 4 2 -> 5
## (Intercept)      1      4      1      4      1     16     16     22     16
## boat.dist        2      5      5     11     14     17      5      5     26
## time             3      6      9     12     12     18     21     24     12
##             2 -> 6 3 -> 1 3 -> 2 3 -> 4 3 -> 5 3 -> 6 4 -> 1 4 -> 2 4 -> 3
## (Intercept)     22      4      1      1      4      1     16     22     16
## boat.dist       29     32     35     38     41     44     47     50     38
## time            12     33     36     39     42     42     48     51     54
##             4 -> 5 4 -> 6 5 -> 1 5 -> 2 5 -> 3 5 -> 4 5 -> 6 6 -> 1 6 -> 2
## (Intercept)     16     22      4      1      4      1      1     16     22
## boat.dist       56     59     62     65     68     68     74     77     80
## time            42     42     63     66     69     72     75     78     81
##             6 -> 3 6 -> 4 6 -> 5
## (Intercept)     16     22     16
## boat.dist       68     68     89
## time            84     87     75
```

Again, `betaCons` constrains any of the transition probability matrix working parameters with the same index to be equal to one another. For example, all of the intercept terms indexed by a '1' ($1 \to 2$, $1 \to 4$, $1 \to 6$, $3 \to 2$, $3 \to 4$, $3 \to 6$, $5 \to 2$, $5 \to 4$, and $5 \to 6$) are equal, and these terms correspond to the transitions from the "ARS" movement mode (states 1, 3, and 5) to the "transit" movement mode (states 2, 4, and 6). Similarly, all of the intercept terms indexed by a '16' are equal and correspond to the transitions from the "transit" movement mode to the "ARS" movement mode. For further details on the `betaCons` argument, see the `fitHMM` help file and the northern fulmar example code in the "vignettes" source directory.

Now we are ready to fit our 6-state HMM:

```
m2 <- fitHMM(fulmar_data, nbStates, dist,
             Par0 = Par0$Par, beta0 = Par0$beta0,
             formula = formula,
             estAngleMean = list(angle = TRUE),
             circularAngleMean = list(angle = TRUE),
             DM = DM, workBounds = workBounds, betaCons = betaCons,
             fixPar = fixPar, knownStates = knownStates,
             stateNames = stateNames)
```

With decent starting values, this model required about 2 min to fit on a standard desktop computer (macOS El Capitan, 2.8 GHz Intel Core i7, 16 GB RAM). For com-

parison, Pirotta *et al.* (2018) required about 18 hr to fit their Bayesian model using MCMC (2.9 GHz Intel Core i7, 16 GB RAM).

We can compare the estimated activity budgets with those of Pirotta *et al.* (2018) using the `timeInStates` function:

```
timeInStates(m2)
```

```
##      seaARS     seaTr   boatARS     boatTr colonyARS  colonyTr
## 1 0.2582469 0.1687088 0.1705938 0.06503299 0.1677663 0.1696513
```

Pirotta *et al.* (2018) estimated 0.28 ("seaARS"), 0.20 ("seaTr"), 0.18 ("boatARS"), 0.06 ("boatTr"), 0.12 ("colonyARS"), and 0.16 ("colonyTr"). While these are very similar, there a handful of state assignments that differ between the analyses. These differences could be attributable to several factors, including: 1) the use of informative priors in the Bayesian analysis of Pirotta *et al.* (2018); 2) our use of fixed trip-level effects on the "sea" state turn angle concentration parameters; and 3) Pirotta *et al.* (2018) assumed the state transition probability covariates ("boat.dist" and "time") only affected the movement direction states ("sea", "boat", "colony"), but in our `momentuHMM` implementation the covariates can affect state transitions for both the movement direction ("sea", "boat", "colony") and the movement mode ("ARS", "transit").

We can also examine activity budgets by individual bird (which are indexed in the raw data "birdID" column), where it is clear that the first 3 individuals tended to spend a larger proportion of their foraging trips in the "boat" states:

```
timeInStates(m2, by = "birdID")
```

```
##   birdID     seaARS       seaTr    boatARS     boatTr  colonyARS
## 1      1 0.35564854 0.117154812 0.22594142 0.11297071 0.09623431
## 2      2 0.14102564 0.134615385 0.28205128 0.10256410 0.25641026
## 3      3 0.08333333 0.269607843 0.35294118 0.06862745 0.04411765
## 4      4 0.49019608 0.196078431 0.00000000 0.04575163 0.05882353
## 5      5 0.61475410 0.008196721 0.00000000 0.00000000 0.30327869
## 6      6 0.00000000 0.235294118 0.05882353 0.02673797 0.32085561
##     colonyTr
## 1 0.09205021
```

```
## 2 0.08333333
## 3 0.18137255
## 4 0.20915033
## 5 0.07377049
## 6 0.35828877
```

Finally, we can create a plot similar to Pirotta *et al.* (2018) using the `plotSat` function (Figure 12):

```
plotSat(m2, zoom = 7, shape = c(17,1,17,1,17,1), size = 2,
        col = rep(c("#E69F00", "#56B4E9", "#009E73"), each = 2),
        stateNames = c("sea ARS", "sea Transit",
                       "boat ARS", "boat Transit",
                       "colony ARS", "colony Transit"),
        projargs = newProj, ask = FALSE)
```

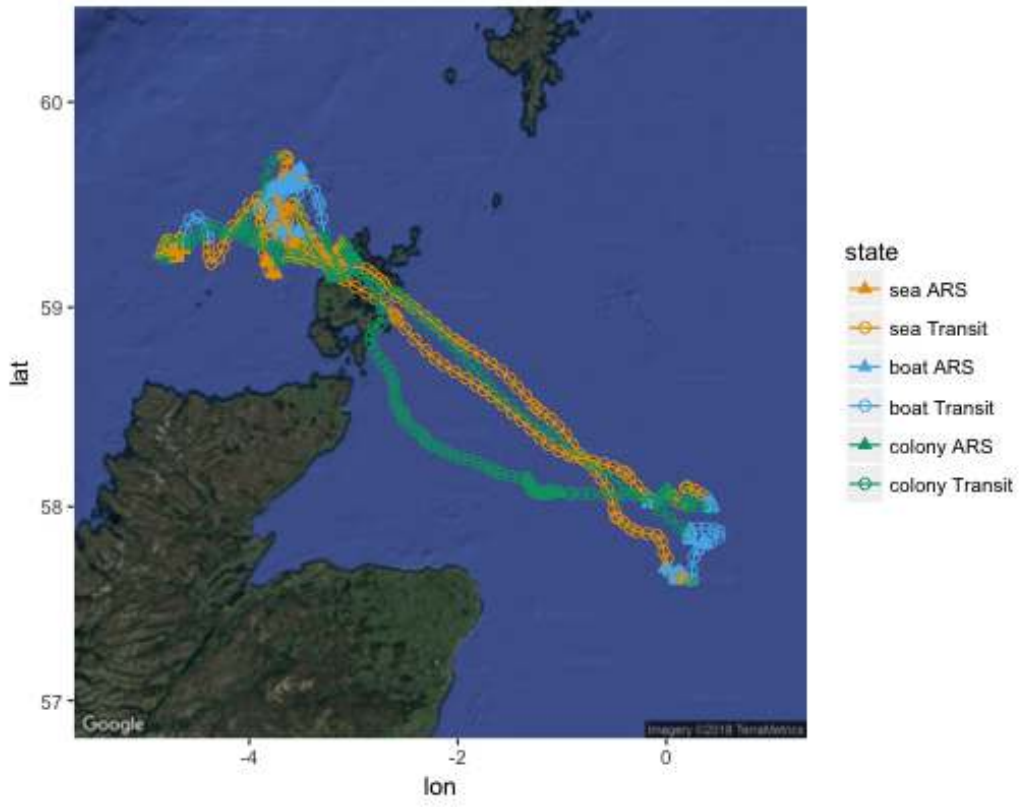

Figure 12. *Seven northern fulmar tracks, colored by the most likely state sequence.*

4 Discussion

Here we have introduced version 1.4.3 of the R package `momentuHMM` and demonstrated some of its capabilities for conducting multivariate HMM analyses with animal location, auxiliary biotelemetry, and environmental data. The package allows for fitting (and simulating from) a suite of biased and correlated random walk movement process models (e.g. McClintock *et al.* 2012), can be used for an unlimited number of data streams and latent behavior states, includes multiple imputation methods to account for measurement error, temporal irregularity, and other forms of missing data that would otherwise be prohibitive to maximum likelihood analysis, and integrates seamlessly with rasters to facilitate spatio-temporal covariate modelling. Because the package incorporates biased random walks, it can also be used to implement group dynamic models (e.g. Langrock *et al.* 2014). The package therefore greatly expands on available software and facilitates the incorporation of more ecological and behavioral realism for hypothesis-driven analyses of animal movement that account for many of the challenges commonly associated with telemetry data. While many of the features of `momentuHMM` were motivated by animal movement data, we note that the package is not limited to location data and can be used for analyzing any type of data that is amenable to (multivariate) HMMs.

Model fitting in `momentuHMM` is relatively fast because the forward algorithm (Eq. 1) is coded in C++. Because multiple imputations are completely parallelizable, with sufficient processing power computation times for analyses that account for measurement error, temporal irregularity, or other forms of missing data need not be longer than that required to fit a single HMM. However, computation times will necessarily be longer as the number of states and/or parameters increase. For example, `momentuHMM` required about 1 hr to fit a single HMM with $N = 6$ states, seven data streams, and $T = 7414$ time steps (McClintock 2017).

As in any maximum likelihood analysis based on numerical optimization, computation times will also depend on the starting values (`Par0` and `beta0`). Specifying "good" starting values is arguably the most challenging aspect of model fitting in `momentuHMM`, particularly for the working scale coefficients when using covariates. The `getPar`, `getPar0`, `getParDM`, and `checkPar0` functions are designed to help with the specification of starting values, and the `retryFits` argument in `crawlWrap`, `fitHMM`,

and `MIfitHMM` will re-optimize based on random perturbations of the parameters to help explore the likelihood surface and diagnose convergence to local maxima. Optimization for the circular-linear regression link function (`tan(mean/2)`; see Table 2) in particular can be prone to local minima, so users are encouraged to explore a range of starting values when fitting these models.

While `momentuHMM` includes functions for drawing realizations of the position process based on the CTCRW model of Johnson *et al.* (2008), this is but one of many methods for performing the first stage of multiple imputation. Realizations of the position process from any movement model that accounts for measurement error and/or temporal irregularity (e.g. Calabrese *et al.* 2016; Gurarie *et al.* 2017) could be passed to `MIfitHMM` for HMM-type analyses in the second stage. Multiple imputation methods also need not be limited to these telemetry error scenarios. For example, conventional missing data could also be imputed using standard techniques (Rubin & Schenker 1986), thereby allowing the investigation of non-random mechanisms for missingness that can be problematic if left unaccounted for in HMMs.

There remain many potential avenues for refining and extending the capabilities of `momentuHMM`. Computation times could likely be improved by further optimizing the R and C++ code for speed. Notable extensions include hidden semi-Markov models and random effects on data stream probability distribution and state transition probabilitiy parameters (Zucchini *et al.* 2016). We would also like to incorporate additional parameters for change-point thresholds and the locations of activity centers instead of requiring that they be pre-specified (and potentially compared using AIC or other model selection criteria) as in grey seal example. Lastly, it is relatively straightforward to add additional probability distributions, and we are pleased to do so upon request. Practitioners interested in additional features for `momentuHMM` are encouraged to contact the authors.

Acknowledgments

We are grateful to R. Scott, B. Godley, M. Godfrey, J. Sudre, and North Carolina Aquariums for providing the data used in our turtle example. The findings and conclusions in the manuscript are those of the author(s) and do not necessarily represent the views of the National Marine Fisheries Service, NOAA. Any use of trade, product,

or firm names does not imply an endorsement by the US Government.

References

Beyer, H.L., Morales, J.M., Murray, D. & Fortin, M.J. (2013) The effectiveness of Bayesian state-space models for estimating behavioural states from movement paths. *Methods in Ecology and Evolution*, **4**, 433–441.

Calabrese, J.M., Fleming, C.H. & Gurarie, E. (2016) ctmm: an R package for analyzing animal relocation data as a continuous-time stochastic process. *Methods in Ecology and Evolution*, **7**, 1124–1132.

Cornelissen, G. (2014) Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling*, **11**, 16.

Costa, D.P., Robinson, P.W., Arnould, J.P., Harrison, A.L., Simmons, S.E., Hassrick, J.L., Hoskins, A.J., Kirkman, S.P., Oosthuizen, H., Villegas-Amtmann, S. *et al.* (2010) Accuracy of argos locations of pinnipeds at-sea estimated using fastloc gps. *PloS one*, **5**, e8677.

DeRuiter, S.L., Langrock, R., Skirbutas, T., Goldbogen, J.A., Calambokidis, J., Friedlaender, A.S. & Southall, B.L. (2017) A multivariate mixed hidden Markov model to analyze blue whale diving behaviour during controlled sound exposures. *The Annals of Applied Statistics*, **11**, 362–392.

Gilbert, P. & Varadhan, R. (2016) *numDeriv: Accurate Numerical Derivatives*. R package version 2016.8-1.

Gurarie, E., Fleming, C.H., Fagan, W.F., Laidre, K.L., Hernández-Pliego, J. & Ovaskainen, O. (2017) Correlated velocity models as a fundamental unit of animal movement: synthesis and applications. *Movement Ecology*, **5**, 13.

Hijmans, R.J. (2016a) *geosphere: Spherical Trigonometry*. R package version 1.5-5.

Hijmans, R.J. (2016b) *raster: Geographic Data Analysis and Modeling*. R package version 2.5-8.

Hooten, M.B., Johnson, D.S., McClintock, B.T. & Morales, J.M. (2017) *Animal Movement: Statistical Models for Telemetry Data*. CRC Press.

Johnson, D.S. (2017) *crawl: Fit Continuous-Time Correlated Random Walk Models to Animal Movement Data*. R package version 2.1.1.

Johnson, D.S., London, J.M., Lea, M.A. & Durban, J.W. (2008) Continuous-time correlated random walk model for animal telemetry data. *Ecology*, **89**, 1208–1215.

Jonsen, I.D., Flemming, J.M. & Myers, R.A. (2005) Robust state–space modeling of animal movement data. *Ecology*, **86**, 2874–2880.

Langrock, R., Hopcraft, G., Blackwell, P., Goodall, V., King, R., Niu, M., Patterson, T., Pedersen, M., Skarin, A. & Schick, R. (2014) Modelling group dynamic animal movement. *Methods in Ecology and Evolution*, **5**, 190–199.

Langrock, R., King, R., Matthiopoulos, J., Thomas, L., Fortin, D. & Morales, J.M. (2012) Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions. *Ecology*, **93**, 2336–2342.

McClintock, B.T. (2017) Incorporating telemetry error into hidden markov models of animal movement using multiple imputation. *Journal of Agricultural, Biological, and Environmental Statistics*, **22**, 249–269.

McClintock, B.T., Johnson, D.S., Hooten, M.B., Ver Hoef, J.M. & Morales, J.M. (2014) When to be discrete: the importance of time formulation in understanding animal movement. *Movement Ecology*, **2**, 21.

McClintock, B.T., King, R., Thomas, L., Matthiopoulos, J., McConnell, B.J. & Morales, J.M. (2012) A general discrete-time modeling framework for animal movement using multistate random walks. *Ecological Monographs*, **82**, 335–349.

McClintock, B.T., London, J.M., Cameron, M.F. & Boveng, P.L. (2017) Bridging the gaps in animal movement: hidden behaviors and ecological relationships revealed by integrated data streams. *Ecosphere*, **8**, e01751.

McClintock, B.T. & Michelot, T. (2018) momentuHMM: R package for generalized hidden Markov models of animal movement. *Methods in Ecology and Evolution*, **9**, 1518–1530.

McClintock, B.T., Russell, D.J., Matthiopoulos, J. & King, R. (2013) Combining individual animal movement and ancillary biotelemetry data to investigate population-level activity budgets. *Ecology*, **94**, 838–849.

Michelot, T., Langrock, R., Bestley, S., Jonsen, I.D., Photopoulou, T. & Patterson, T.A. (2017) Estimation and simulation of foraging trips in land-based marine predators. *Ecology*, **98**, 1932–1944.

Michelot, T., Langrock, R. & Patterson, T.A. (2016) moveHMM: An R package for the statistical modelling of animal movement data using hidden Markov models. *Methods in Ecology and Evolution*, **7**, 1308–1315.

Morales, J.M., Haydon, D.T., Frair, J., Holsinger, K.E. & Fryxell, J.M. (2004) Extracting more out of relocation data: building movement models as mixtures of random walks. *Ecology*, **85**, 2436–2445.

Pirotta, E., Edwards, E.W.J., New, L. & Thompson, P.M. (2018) Central place foragers and moving stimuli: A hidden-state model to discriminate the processes affecting movement. *Journal of Animal Ecology*, **87**, 1116–1125.

R Core Team (2017) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rivest, L.P., Duchesne, T., Nicosia, A. & Fortin, D. (2016) A general angular regression model for the analysis of data on animal movement in ecology. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **65**, 445–463.

Rubin, D.B. & Schenker, N. (1986) Multiple imputation for interval estimation from simple random samples with ignorable nonresponse. *Journal of the American Statistical Association*, **81**, 366–374.

Sarda-Espinosa, A. (2017) *dtwclust: Time Series Clustering Along with Optimizations for the Dynamic Time Warping Distance*. R package version 3.1.2.

Wall, J., Wittemyer, G., LeMay, V., Douglas-Hamilton, I. & Klinkenberg, B. (2014) Elliptical time-density model to estimate wildlife utilization distributions. *Methods in Ecology and Evolution*, **5**, 780–790.

Whoriskey, K., Auger-Méthé, M., Albertsen, C.M., Whoriskey, F.G., Binder, T.R., Krueger, C.C. & Mills Flemming, J. (2017) A hidden markov movement model for rapidly identifying behavioral states from animal tracks. *Ecology and Evolution*, **7**, 2112–2121.

Zucchini, W., MacDonald, I.L. & Langrock, R. (2016) *Hidden Markov Models for Time Series: An Introduction Using R*. CRC Press.