Alternative parametric considerations for direction and distance when modelling animal movement

Gopika Ramkilawon 16082584

Submitted in partial fulfillment of the degree MSc Advanced Data Analytics

Supervisors: Prof J. T. Ferreira, Dr N. Nakhaeirad

Department of Statistics, University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Declaration

I, *Gopika Devi Ramkilawon,* declare that this essay, submitted in partial fulfillment of the degree *MSc Advanced Data Analytics,* at the University of Pretoria, is my own work and has not been previously submitted at this or any other tertiary institution.

_____

Gopika Devi Ramkilawon

_____

Prof J.T. Ferreira

_____

Dr N. Nakhaeirad

_____

February 4, 2022

Contents

List of Figures

List of Tables

Table of abbreviations

A table summarising the abbreviations used throughout this report is presented below with the corresponding definition in order of appearance.

Abbreviation	Definition
GPS	global positioning system
GIS	geographic information system
HMM	hidden Markov model
pdf	probability density function
BCRW	biased correlated random walk
cdf	cumulative distribution function
IMM	independent mixture models
SSF	step selection function
IBM	individual based model
EM	expectation maximisation
ssvM	sine-skewed von Mises
EC	Exponentiated Cardioid
mgf	moment generating function
AIC	Akaike information criterion
BIC	Bayesian information criterion
CRW	correlated random walk
BRW	biased random walk
DBN	dynamic Bayesian network
MRW	multi-state random walk

List of Notation

A list summarising all notation used is presented below within subsections of their relevant topics.

Data Notation

T	the total number of observed animal locations
x_{jt}	the value of the j-th explanatory directional variable
z_{jt}	the value of the j-th explanatory real variable
y_t	the direction between the animal's position from step t to step $t+1$
$y_{0:T}$	the set of all observed directions $y_0, \ldots, y_T$
d_t	the distance between the animal's position from step t to step $t+1$
$d_{0:T}$	the set of all observed distances $d_0, \ldots, d_T$
$\mathscr{F}_t^o$	the observed data filtration containing directions,distances and explanatory variables from time 0 till time t

Hidden Markov Process

S_t	the state the animal is in at time t
$S_{0:T}$	the set of all states $S_0, \ldots, S_T$
S_{kt}	An indicator function which equals 1 when $S_t = k$ and 0 if $S_t \neq k$
$\mathscr{F}_t^c$	the complete information filtration which contains all of the observed and hidden state information from $S_{0:T}$
$g(S_t \vert \mathscr{F}_{t-1}^c)$	the conditional pdf of the hidden state S_t
π_{rk}	the state transition probability denoted by $P(S_t = k \vert S_{t-1} = r)$

Observed Animal Trajectory

$k(y_t, d_t \vert S_t, \mathscr{F}_{t-1}^c)$	the conditional joint pdf of the observed data of y_t and d_t
$f(y_t \vert S_t, \mathscr{F}_{t-1}^c)$	the conditional pdf of y_t
$h(d_t \vert S_t, \mathscr{F}_{t-1}^c)$	the conditional pdf of d_t
$f_k(y_t \vert \mathscr{F}_{t-1}^c; \kappa^{(k)})$	the pdf of y_t given that the hidden Markov state is k at time step t
$\kappa^{(k)}$	the vector of parameters of f_k
$\mu_t^{(k)}$	the mean direction of the pdf f_k
$\ell_t^{(k)}$	the concentration parameter of f_k
$h_k(d_t, \phi^{(k)})$	the pdf of d_t given that the hidden Markov state is in state k at time step t
$\phi_1^{(k)}, \phi_2^{(k)}$	the parameters of h_k

Abstract

Animal movement is a fundamental part of ecology, and aids in understanding and modelling of social responsibility phenomena including population and community structure dynamics. Movement of animals is often characterised by direction (measured on the circle) and distance (measured on the real line); but traditional employed models often do not account for potential asymmetric directional movement, or departures from the usual gamma or Weibull assumptions for distance. This study focuses on the modelling of circular data in this animal movement environment on previously unconsidered circular distributions such as the sine-skewed von Mises distribution which may allow and account for departures from symmetry. In addition, alternative models to the aforementioned gamma or Weibull assumptions for distance are considered, namely the power Lindley (as a mixture of gamma and Weibull) as well as a Gumbel candidate. Computational aspects and investigations of this joint modelling is highlighted, particularly via the illustration of an extensive bootstrap study. A general hidden state Markov model is used to incorporate both these essential components when estimating via the use of the EM algorithm, and goodness of fit measures verifies the validity and viable future consideration of the newly proposed theoretical models within this practical and computational animal movement environment. The ethics number for this study is NAS124/2019.

Acknowledgements

The support of the DSI-NRF Centre of Excellence in Mathematical and Statistical Sciences (CoE-MaSS) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the CoE.

This work formed part of the grant RDP 296/2019 based at the University of Pretoria.

I would like to thank both of my supervisors, Prof Johan Ferreira and Dr Najmeh Nakhaeirad for their support, assistance, patience and tolerance with me throughout my research journey. Without your guidance, I certainly would not have found my direction in my research journey within animal movement modelling. Your vast and expert insights in the field of statistics have been a pool of gold for me and have motivated me to continuously learn and sharpen my skills. I am deeply grateful to have the both of you as my supervisors and I am honoured to work with the both of you.

Finally, I would like to thank Prof Andriette Bekker for providing me with the opportunity of pursuing my statistical journey with the University of Pretoria. I am deeply grateful for the opportunities which have been granted to me within the Department of Statistics in both my undergraduate and postgraduate studies.

1 Introduction

In ecology, the understanding and relationship between animal movement and their respective habitat heterogeneity (including the classification and characterisation of various strategies used by animals to locate sites for safety and forage) are difficult. The understanding of these movement and behavioural patterns with consideration of the consequences of climate change, pollution, land use and the proliferation of other invasive species and diseases in the manner of the motion of hosts and their pathogens into new impuissant areas [40]. A major challenge in the modelling of animal movement with consideration to various environmental features is in the validation of a suitable statistical model and approach since the true and exact behaviours of animals are rarely known, despite technological advancements in animal tracking.

Modelling animal movement and locations provide fundamental insights into mechanisms behind the distribution and behaviours of the particular animal. Originally, animal data collection was performed manually on foot by using fieldwork. Various technological advancements, such as the development of the Global Positioning System (GPS) and geographic information systems (GIS) permit the collection of large amounts of animal movement data with a certain degree of accuracy. The collection of this data then allows researchers to investigate the influence of the animal's environment on the animal's displacement. These developments provide more accurate readings of movement and displacement which are not limited to logistics or sunlight, as well as images and footage of the animals in their respective habitats without a human risk factor, which was prevalent when manual recordings were originally performed.

There is no single universally known accepted method for the inference of the behavioural states (such as resting", "foraging" and "travelling") which generate the recorded geographical location of animals. To model animal movement, various robust statistical techniques and pliable animal movement models are required [32]. Within an equally-spaced discrete time environment, displacement may be characterised by the direction and distance between two consecutive points with a circular-linear process modelling the movement within this two-dimensional environment. A popular example of models used for animal movement are Hidden Markov Models (HMMs) since they are flexible and practical to segment movement pathways into latent behavioural states [15] of the animal's movement. Other commonly used models include variants of random walk models, general linear mixed models and independent mixture models.

Within this environment of animal movement models, directional movement and distance of the animal is of utmost importance. This holds true for many animal movement models utilising hidden states to classify the states of movement for the animal being investigated. Traditional models have included the Gamma and Weibull distributions for distance modelling in conjunction with the von Mises distribution for directional modelling. However, these models are sometimes limited in their flexibility and use in terms of accounting for potential

irregular movement patterns in both direction and distance. In this study, alternative parametric considerations for distribution fitting for both distance and directional variables within the animal movement focus area within statistical modelling are developed, proposed, and implemented.

1.1 Directional Models

In this section, core elements of statistical distribution theory and modelling aspects are reviewed relevant to directional models and how they may be evaluated. The directional model includes the von Mises distribution, which is popularly used in animal movement models to measure directional movement. In this section, the variable θ denotes the direction in which the animal is turning whilst travelling.

1.1.1 Circular Models

Circular data is data which is measured on a unit circle in degrees or radians [10] which has a periodic nature. Each circular observation may be considered as a point on a circle with a unit radius or vector in a plane $\mathbb{R}^n$. Furthermore, each observation may be specified by the angle being measured from the initially chosen direction to the point on the circle corresponding to the relevant observation [28].

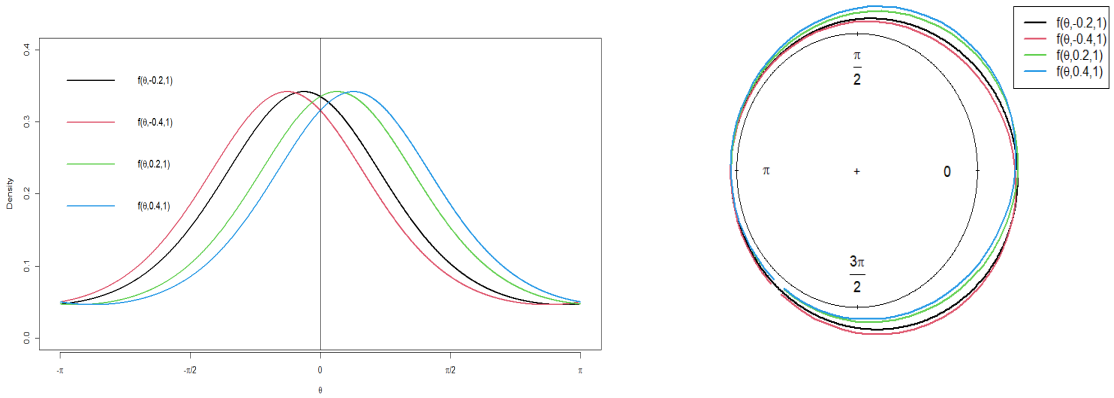
A popular circular distribution is the von Mises distribution which is denoted by $M(\mu, \kappa)$ and has a pdf (probability density function) [28]:

$$g(\theta; \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp(\kappa \cos(\theta - \mu)), \quad -\pi \leq \theta \leq \pi \tag{1}$$

where I_0 denotes the modified Bessel function of order 0 of the first kind (86).

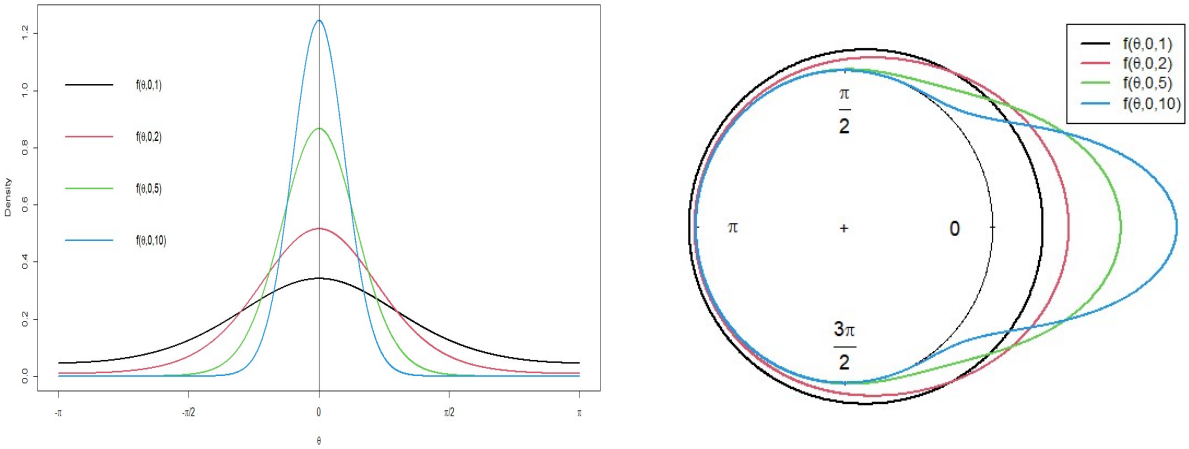
The mean direction of the $M(\mu, \kappa)$ is denoted by $\mu \in [-\pi, \pi)$ and the concentration parameter is denoted by $\kappa \geq 0$. The von Mises distribution has a core role among other circular distributions and is often referred to as the circular normal distribution. The variable θ is the directional deviation of the observed point from the mean of the distribution [16].

The pdf of $M(\mu, \kappa)$ (1) is illustrated below for various values of μ and κ:



(a) The pdf of $M(\mu, \kappa)$ (b) The circular curve of $M(\mu, \kappa)$

Figure 1: The pdf and circular curve of $M(\mu, \kappa)$ for $-\pi \leq \theta \leq \pi$ for different values of μ



(a) The pdf of $M(\mu, \kappa)$ (b) The circular curve of $M(\mu, \kappa)$

Figure 2: The pdf and circular curve of $M(\mu, \kappa)$ for $-\pi \leq \theta \leq \pi$ for different values of κ

The shifts in values for μ and κ have different effects on the shape of the pdf and circular curve of $M(\mu, \kappa)$. An increase in the value of μ results in the pdf shifting to the right whereas a decrease in μ results in a shift to the left in the pdf. As the value of κ increases, the peak of the pdf increases and as it decreases, the pdf becomes flatter and the peak decreases.

The von Mises distribution may be expressed with a mean direction μ_t depending on μ_{t-1} and other explanatory angles within circular regression models for biased correlated random walks (BCRW) which are introduced in Section 5.2.3. This mean direction also depends on a homogenous error whose distribution depends on the aforementioned concentration parameter κ. The von Mises is a favoured distribution due to its mathematical tractability which can be implemented into circular regression models with relative analytic and computational

ease. However, the symmetric, unimodal nature of the von Mises may be a limitation in modelling, particularly in scenarios where the angular movement is skewed and multimodal. This would require a more flexible angular distribution which is able to account for skewness and asymmetry.

Circular models can be quite challenging to implement into movement models. This is due to the complex form structure and high mathematical complexity of its pdf and corresponding moments. Some circular models do not have closed forms, making modelling and implementation quite difficult. Furthermore, multimodality in the log-likelihood for estimating these parameters is common, making the optimisation for global maxima challenging; so various alternative estimation procedures other than maximum likelihood need to be used.

Extensions of the von Mises distribution to model higher dimensions is obtained by expressing (1) in Cartesian co-ordinates. An observation is then represented by a unit vector $\mathbf{x}$ and the corresponding mean is represented by a unit vector denoted by $\boldsymbol{\nu}$. Then, (1) becomes [16]

$$g(\theta; \boldsymbol{\nu}, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp(\kappa \boldsymbol{x'\nu}), \quad -1 \leq \boldsymbol{\nu} \leq 1. \tag{2}$$

In the two-dimensional case the unit vector is defined as, $\boldsymbol{\nu'} = [\cos(\mu), \sin(\mu)]$ and for three-dimensional case, the unit vector is defined as, $\boldsymbol{\nu'} = [\cos(\phi), \sin(\phi)\cos(\mu), \sin(\phi)\sin(\mu)]$ [16]. The ϕ is the angular (directional) deviation of the mean vector from the 'North Pole' defined by the co-ordinate system, called the 'colatitude' and μ is the longitude of the mean vector.

1.2 Distance Models

In this section, core elements of statistical distribution theory and modelling aspects is reviewed regarding distance models and how they may be evaluated. This distance refers to the distance travelled by the animal between consecutive time steps i.e. their step lengths. To model this distance, any pdf on the positive real line may be considered. Both distributions on the next page have a shape and scale parameter which makes them ideal candidates for consideration. The shape and scale parameters give the distributions some unique characteristics, however, may not always be suitable across all scenarios. For this reason, many other pdfs with different parameter definitions may be considered and compared to the performance of the Gamma and Weibull distributions. These distributions have been considered in literature by Nicosia et al. [32], Van Niekerk [40] and Langrock et al. [24].

1.2.1 Gamma Distribution

The Gamma distribution is a continuous distribution which is popularly used in statistics. A continuous variable X has the Gamma distribution with parameters $\kappa > 0$ and $\theta > 0$ if it has the pdf [3]

$$f(x;\theta,\kappa) = \frac{1}{\theta^{\kappa}\Gamma(\kappa)} x^{\kappa-1} \exp\left(-\frac{x}{\theta}\right), \quad x > 0 \tag{3}$$

where $\Gamma(\cdot)$ denotes the gamma function (91) (see Appendix). If X has this pdf, it is denoted $X \sim$GAM(θ,κ). The shape parameter is denoted by κ since it determines the shape of the pdf graph, dependent on the value of κ. When $\kappa < 1$, the pdf exponentially decreases rapidly until it tends to zero. When $\kappa = 1$, the exponential decrease is much slower than when $\kappa < 1$. When $\kappa > 1$, the pdf initially increases before slowly exponentially decreasing towards zero.

The parameter θ denotes the scale parameter for the Gamma distribution. This parameter is of importance because this ensures that the results will not be dependent on the scale of measurement used [3]. The cumulative distribution function (CDF) of the Gamma distribution is [3]

$$F(x;\theta,\kappa) = \int_0^x \frac{1}{\theta^{\kappa}\Gamma(k)} p^{\kappa-1} \exp\left(-\frac{p}{\theta}\right) dp. \tag{4}$$

Below, a table summarising the moments of the Gamma distribution is presented [3]:

Expected Value	E$(X) = \kappa\theta$
Variance	Var(X)=$\kappa\theta^2$
Moment Generating Function	$M_X(t) = (1-\theta t)^{-k}$ for $t < \frac{1}{\theta}$

The maximum likelihood estimators of the Gamma distribution parameters (θ and κ), are respectively given by [3]:

$$\hat{\theta} = \frac{\bar{x}}{\hat{\kappa}} \tag{5}$$

where $\bar{x}$ denotes the sample mean from n considered observations and

$$\ln(\hat{\kappa}) - \frac{\Gamma'(\hat{\kappa})}{\Gamma(\hat{\kappa})} - \ln\left(\frac{\bar{x}}{(\prod_{i=1}^{n} x_i)^{\frac{1}{n}}}\right) = 0 \tag{6}$$

Note that the maximum likelihood estimator for κ does not have a closed form, so iterative procedures are used for estimation.

1.2.2 Weibull Distribution

The Weibull distribution is a continuous distribution popularly used as a failure-time distribution. A continuous variable X has the Weibull distribution with parameters $\beta > 0$ and $\theta > 0$ if it has the pdf [3] with the form

$$f(x, \theta, \beta) = \frac{\beta}{\theta^{\beta}} x^{\beta-1} \exp\left(-\left(\frac{x}{\theta}\right)^{\beta}\right), \quad x > 0 \tag{7}$$

and zero otherwise. If X has this pdf, it is denoted $X \sim$Wei(θ, β). The shape parameter is denoted by β since it determines the shape of the pdf graph, dependent on whether $\beta < 1$, $\beta = 1$ or $\beta > 1$. Furthermore, there are various asymptotes associated with the choice of β, influencing the range of y values for the pdf. The parameter θ is known as the scale parameter. The CDF of the Weibull distribution is [3]

$$F(x; \theta, \beta) = 1 - \exp\left(-\left(\frac{x}{\theta}\right)^{\beta}\right), \quad x > 0. \tag{8}$$

Below, a table summarising the moments of the Weibull distribution is presented [3]:

Expected Value	$\mathsf{E}(X) = \theta\Gamma\left(1 + \frac{1}{\beta}\right)$
Variance	$\mathsf{Var}(X) = \theta^2\left\{\Gamma\left(1 + \frac{2}{\beta}\right) - \Gamma^2\left(1 + \frac{1}{\beta}\right)\right\}$
Moment Generating Function	$M_X(t) = \sum_{n=0}^{\infty} \frac{t^n \theta^n}{n!}\Gamma\left(1 + \frac{n}{\beta}\right)$, $\beta \geq 1$

The maximum likelihood estimators of the Weibull distribution parameters (θ and β), are respectively given by [3]:

$$\hat{\theta} = \left(\frac{\sum_{i=1}^{n} x_i^{\beta}}{n}\right)^{\frac{1}{\beta}} \tag{9}$$

and the solution for the estimate of $\hat{\beta}$ is the solution of the following equation

$$\frac{\sum_{i=1}^{n} x_i^{\beta} \ln(x_i)}{\sum_{i=1}^{n} x_i^{\beta}} - \frac{1}{\beta} - \frac{\sum_{i=1}^{n} \ln(x_i)}{n} = 0. \tag{10}$$

Note that (10) cannot be solved explicitly and would also require iterative procedures for estimation.

1.3 Background Literature On Animal Movement

Within the animal movement atmosphere, a variety of authors have developed animal movement models each with their advantages and drawbacks. These models include individual-based models [25], multi-level models with a generalised linear mixed model as the basis [29], advection-diffusion models [12], hidden Markov models (HMMs) with independent mixture models [15], general hidden-state random walks [32], step selection function models

(SSF) [33] and biased correlated random walk (BCRW) models [11]. The selection of a particular model is based on multiple factors, such as the aims and objectives of the study together with the chosen animal movement data the researcher wishes to use for the application of the modelling process.

The wide variety of models already used by various authors, therefore, makes it an arduous task to design a model which encompasses all the features of the animal's environment and the movement of the animal within its particular landscape whilst maintaining an interpretable and computationally efficient model. There have been a limited amount of models which account for several movement taxis concerning features of the environment [32]. Furthermore, models which are sufficiently flexible to account for multimodality and skewness are uncommon in the animal movement focus area. These are common limitations in many models which should be addressed. However, the construction and fitting process for models which can account for the aforementioned limitations typically become more computationally demanding which is a drawback of mathematically complex modelling [32].

Many factors such as heterogeneity in the animal's movement, step length, step angle distributions and measurement errors in locations need to be addressed, which makes fitting a uniquely specified animal movement model an arduous task [32]. However, with technological advancements, modelling improvements and further research endeavours, implementation of such a model to satisfy the aims, objects and problem of this study would be eased. Potential advantages and disadvantages of the model and its implementation will be critically evaluated and compared with results obtained by other authors.

Latombe et al. (2014) designed an individual based model (IBM) [1] of a caribou's movement that are based on mechanisms which are generative at low emergence levels and which guarantees the model's ability to generalise accordingly. Forwarding mechanisms were used to parametrise the IBM's trait, enabled by the artificial production of surrogate data for variables at the low levels of emergence which are not accessible from the GPS data obtained. This procedure enabled Latombe et al. (2014) to ensure that the IBM was statistically relevant to explain the data as well as have identifiable parameters. The usage of the statistical mode as a feature of the IBM increased the robustness of the model which therefore enabled the usage of k-fold cross-validation and emergent patterns validation as two validation processes which are independent of each other. The advantage of using forward modelling was the reduction of iterations required for computational ease. This system permitted a model design that encompassed the complexity of the system as well as projections on future possible states which react to different management plans. This ensured relevance in long-term scenario impact testing corresponding to unobserved environmental configurations.

Moreau et al. (2012) made use of multi-level functional responses, where generalised linear mixed models formed the modelling base for twenty-seven threatened forest-dwelling female caribou within and among certain geographical home-ranges. It was illustrated that these functional responses may occur across multiple levels,

[1] IBM's are an alternative predictive models which are able to easily integrate internal states in their relevant implementations [25]

including nested echelons which adequately explained the plasticity in the observed habitat. Moreau et al. (2012) discovered that there was a simultaneous response to both local and global human-related pollution disturbances which provided evidence of a nested-echelon functional response. Their model indicated that the fixing of habitat related requirements based on the habitat selection patterns detected were misleading due to the fact that it overlooked malleability in the animals' responses to habitat heterogeneity. The study demonstrated the importance of the assessment of multi-level functional responses to explain and account for spatial dynamics and to evaluate habitat patches losses for the caribou with human induced habitat loss [29].

Goodall et al. (2017) made comparisons between the Hidden Markov model (HMM) and independent mixture models (IMM) using simulated and field data. While both methods have shown to provide interpretable results, performance varied based on the data. The field data used was of sable antelope movement and African buffalo in the Kruger National Park. All models were fitted and evaluated using R statistical software. Two methods of comparison were used to formally compare the results of applying the two different models to the animal movement data. The simulated data was obtained from the log-normal distribution with the parameters made identical to that of the field data. The HMM was used as the true model with parameters assumed to be known with the true states and the associated observed movements simulated from the true model. The HMM and IMM were fitted to the simulated displacements with their underlying states predicted using the Viterbi algorithm and the mixture likelihood clustering method respectively [15]. Comparisons were drawn using the state classification accuracy of the model and the precision of parameter estimates comparing the known and estimated parameters and states. Parameter comparisons were drawn using 90% confidence intervals and bootstrap standard errors of the estimated parameters. This process was completed for state dependent distribution parameters for both methods as well as the transition probabilities of the HMM [15]. The results indicated that the HMM model consistently obtained confidence intervals which were narrower around parameters with smaller standard errors than the IMM's. However, for some of the data, the improvement shown by the HMM was marginal, indicating that the IMM provides an capable alternative for the identification of an animal's latent behavioural states. A general expectation in a statistical perspective is for HMM's to provide a superior balance between model extensibility and complexity for animal movement modelling, but as shown by Goodall et al. (2017) the IMM could be an acceptable alternative and possibly be more steadfast biologically.

Nicosia et al. (2017) propose a general hidden multi-state random walk model to describe animal movement with a specific application to the movement of caribou in Canada. This proposed model takes movement behavioural patterns with respect to environmental features into account using a generalisation of the biased correlated random walk (BCRW) model where the mean direction depends on several directional targets. The direction and distance between two consecutive locations is modelled by a circular-linear process and a hidden Markov process accounts for changes in animal movement behaviours. This is achieved by more flexible hidden Markov state models which

can account for directional persistence and the influence of multiple environmental targets varying across states [32]. The EM algorithm tailored for this model was included which enabled the prediction of the hidden states of the model and aided in the understanding of the roles of specific targets on the overall animal movement and movement patterns. A simulation study was initially performed based on data simulating the movement of a single animal in the plane. For this simulation study, a two-state Markov chain was generated, where the locations of the animals was obtained by simulating a direction from a von Mises model and distance from a Markov switching model. Two different scenarios with varying directional persistence and repulsion were used for the simulation study. The application on the caribou movement was evaluated using two models; one with a Markov specification for the hidden states and the other with a semi-Markov specification for the hidden states. The distances travelled by the caribou were fitted using the gamma and Weibull distributions. The respective direction i.e. turning angles are von Mises distributed with a specified mean direction and concentration parameter. An important distinction to note in this model is that the model has two states, including directional persistence. The directional analysis performed by Nicosia et al. (2017) identified two environmental features influencing the caribou movement. Furthermore, it was discovered that directional persistence is only important when the animal is travelling between sites. The conclusion reached by Nicosia et al. (2017) was that the proposed model describes animal movement going towards a target well, specifically if the animal is attracted by two targets and if directional bias is towards the closest target at a given step. This clearly improves on classical BCRW since this model permits several movement behaviours of the animal [32].

Nicosia et al. (2017) proposed the Step Selection Function (SSF), which is a multi-state implementation of a method which is based on a conditional logistic regression model which is used to describe animal movement. The SSF is developed from comparing the observed animal location and other randomly sampled locations at each time step. The SSF improves on a classical multi-state BCRW modelling since it allows a global movement strategy and a discrete local habitat selection to evolve over time. The multi-state BCRW only considers the former [33]. Nicosia et al. (2017) then proved that multi-state BCRW can be fitted using the multi-state SSF, which is further validated in a simulation study and in the analysis of a real dataset. The simulation study performed included one target which was placed in the centre of the map, which included a time-homogenous two-state Markov chain to generate the transition matrix, as well as consensus von Mises model for the turning angles with explanatory angles simulated from a gamma distribution, using the EM algorithm for model implementation with a filtering smoothing algorithm. The specific scenario for this simulation study was that the animal demonstrated high directional persistence and high attraction to the target in state 1, and high directional persistence and moderate repulsion from the target in state 2. This simulation study illustrated that the SSF recovers well to the corresponding parameters of the BCRW model, and has the opposite effect in the case where the control locations is a small number [33]. The real dataset used was on the trajectory of an individual bison in Prince Albert National Park in Canada. The

experiment distinguished between the two states and that movement and selection parameters can vary between the two states [33]. The model was also able to identify bison foraging areas and popular trajectories when the bison were moving between these areas. This enabled a few more significant movement behaviour findings in their relevant movement states which highlighted certain behavioural and movement characteristics. Furthermore, the association between the habitat features and exploratory mode gives further insight on landscape connectivity which includes structural and functional components [33]. The findings also found that the proposed model obtains nearly identical estimates between the multi-state general random walk model and the multi-state SSF model.

Duchesne et al. (2015) demonstrated that the log-likelihood function used for the estimation of the parameters of the BCRW model can be approximated using the log-likelihood of SSFs. The relationship between the SSF and BCRW models were illustrated by fitting the BCRW with maximum likelihood estimation and with the SSF to model simulated movement data to the trajectory of bison trials in natural landscapes. Fitting a BCRW and performing SSF analysis characterised both various movement taxes and ecology [11]. To initially estimate the parameter β of the BCRW, the directional circular regression model and the consensus model circular regression model were considered. The von Mises model was used to model the step angles for both the directional model and consensus model. The simulation study was performed to compare the three methods, which are the directional estimation method, consensus estimation method and SSF estimation method which involved two independent animals moving on a landscape. This method of simulation avoided multicollinearity issues when simulating movement for a single animal [11]. Three separate simulations were performed, each consisting of different landscapes and characteristics. The results of the simulation study indicated that the estimators for the SSF and consensus model were very similar, but the SSF estimator was slightly more variable. However, all three methods used estimate β well, with small differences within each of the three simulations performed and each with their own advantages and disadvantages. Another study was performed on bison trial data which was collected in Prince Albert National Park, Canada. The analysis considered only movement taxes and directional persistence. The estimates of the three methods are quite similar. The lack of differences between estimates might have been due to a small amount of controls used for this particular experiment. The study therefore concludes that the BCRW and SSF can result in similar inferences on animal movement with a broad set of outlined conditions. It was also proven that when step angles follow a von Mises consensus BCRW model, using maximum likelihood estimation of the parameters is equivalent to the estimation of the parameters using an SSF model with appropriate covariates and a large number of control step angles which are randomly sampled from a uniform distribution [11]. Furthermore, the study found that both SSF and BCRW are both useful tools for the identification of factors which control movement decisions. The ease at which the SSF may be estimated using many statistical packages makes it versatile and useful to use, especially since many covariates may be simultaneously considered, accelerating the acquisition of information on the determinants of animal movement and their distributions.

1.4 Aims and Objectives

The aims for this study is focused firstly on the goal is to introduce more flexible directional and distance models within this existing Markovian framework which are able to accommodate the asymmetric and skewed directional movement and observations on distance that potentially fits better under other real parametric considerations. Secondly, the generalisation of the BCRW model to ensure that the mean direction depends on several directional targets by embedding the directional model within the BCRW and whilst retaining interpretability will be addressed. The EM algorithm is used to perform model fitting, enabling the posterior probabilities to be calculated for the hidden states at each animal trajectory step.

The objectives of this study are as follows:

1. Fit a uniquely specified Markov switching model to various animal movement datasets with identifiable parameters with a Markovian hidden process.

2. Use a circular regression model to model the direction i.e. step angle using the von Mises and sine-skewed von Mises distributions.

3. Model the distances travelled using the Gamma, Weibull, power Lindley and Gumbel distributions.

4. Implement and fit the model using the EM algorithm and a filtering-smoothing algorithm for the hidden Markov specification.

5. To validate the multi-state circular-linear process previously fitted by Nicosia et al. [32] on the movement of caribou in Canada's boreal forest by drawing comparisons on the estimates and new distributions to the previously obtained results.

1.5 Outline of Study

In Section 2 the new directional and distance models are introduced and discussed. In Section 3 the model and its modelling results are presented. In Section 4 the modelling results and report is concluded with special mention of possible future research endeavours. In Section 5 (Appendix A) important theoretical results and background theory of statistical concepts are introduced and discussed. In Section 6 (Appendix B) the code used for this model is included.

2 Alternative Considerations for Direction and Distance

Previously considered models were discussed in Section 1. In this section, alternative considerations are introduced. Firstly, the directional models are introduced and thereafter, the distance models are introduced.

2.1 Direction

2.1.1 Sine-skewed von Mises Distribution

Classical circular models such as the wrapped Cauchy, wrapped normal and von Mises are subject to limitations such as uni-modality and symmetry. The consideration and development of more flexible models which are not restricted by such limitations is of interest.

Two methods that lead to distributions being capable of modelling asymmetry and potentially multimodality involve the von Mises distribution. The first approach is symmetry modulation of the von Mises pdf, with the second using mixtures of the von Mises distribution [1].

A popular method of creating skew-symmetric models is that of perturbation. The pdf of a random circular variable Θ with corresponding location parameter $-\pi \leq \eta \leq \pi$ is denoted by [1]:

$$f(\theta) = 2f_0(\theta - \eta)G_0(w(\theta - \eta)), \quad -\pi \leq \theta < \pi.$$

This is of importance in obtaining the expression for the sine-skewed von Mises (ssvM) pdf. In the general approach, $f_0(\cdot)$ and $g_0(\cdot)$ are base circular pdfs which are symmetric around zero, with $G_0(\theta) = \int_{-\pi}^{\theta} g_0(\phi)d\phi$ the cdf corresponding to $g_0(\cdot)$ [1]. The weighting function denoted by w is an odd periodic function, which dictates that $w(-\theta) = -w(\theta)$ and $w(\theta) = w(\theta + 2\pi z)$ for all integer values z, such that $|w(\theta)| \leq \pi$.

The issue of mathematical tractability in terms of how components are chosen arises with this construction. This was addressed by using a wrapped Cauchy distribution and von Mises pdf for $f_0(\cdot)$ because of their appealing closed form. The two other components make use of the cdf of the circular uniform distribution with the following form [1]:

$$G_0(\theta) = \frac{\pi + \theta}{2\pi},$$

as well as the weighting function $w(\theta) = \lambda\pi\sin(z\theta)$. In the weighting function, the parameter $\lambda \in [-1, 1]$ is the skewing parameter and z is a positive integer value. These two choices correspond to perturbing f_0 as [1]:

$$f(\theta) = f_0(\theta - \eta)[1 + \lambda\sin(z(\theta - \eta))]. \tag{11}$$

This pdf is left-skewed when $\lambda > 0$, right-skewed when $\lambda < 0$ and is unchanged when $\lambda = 0$. The results

$f(\eta-\theta;\lambda)=f(\eta+\theta;\lambda)$ and $f(\eta)=f_0(0)$ hold regardless of the value of λ [1]. Furthermore, the two endpoints coincide, i.e. $f(\eta-\pi)=\lim_{\theta\to\eta+\pi}f(\theta)$. When $z\geq 2$, the pdf $f(\cdot)$ will be multimodal. In the case where $z=1$, (11) simplifies to:

$$f(\theta)=f_0(\theta-\eta)(1+\lambda\sin(\theta-\eta)). \tag{12}$$

Any pdf obtained using (12) is referred to as a sine-skewed distribution. The cdf $F(\theta)$ of any distribution with pdf (12) can be expressed as [1]:

$$F(\theta)=\int_{-\pi}^{\theta}(1+\lambda\sin(\phi))f_0(\phi)d\phi=F_0(0)+\lambda\int_{-\pi}^{\theta}\sin(\phi)f_0(\phi)d\phi, \tag{13}$$

where $F_0(\theta)=\int_{-\pi}^{\theta}f_0(\phi)d\phi$ is the cdf of the base symmetric distribution with pdf $f_0(\theta)$.

To obtain the ssvM pdf , $f_0(\theta)$ is substituted with the von Mises pdf in (12). The pdf of the ssvM distribution is denoted by [1]:

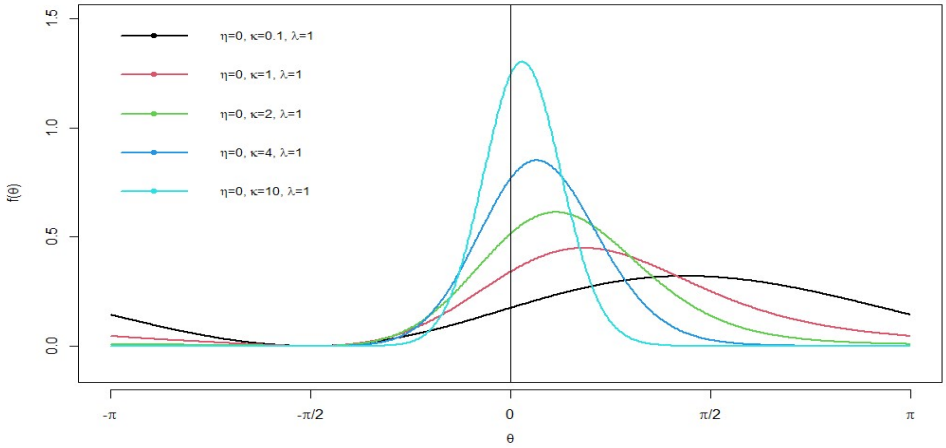
$$f(\theta)=\frac{\exp(\kappa\cos(\theta-\eta))}{2\pi I_0(\kappa)}(1+\lambda\sin r(\theta-\eta)),\quad -\pi\leq\theta<\pi,\ \ -1\leq\lambda\leq 1, \tag{14}$$

where $\kappa\geq 0$ is the concentration parameter, $-\pi\leq\eta<\pi$ is the location parameter, $\phi\in[-1,1]$ is the skewness parameter and $I_0(\kappa)$ is the modified Bessel function of the first kind of order 0 with form (86). When $r=1$, the ssvM pdf may be both unimodal and bimodal depending on the parameter choices. However, when $r>1$, the value of r will then represent the number of modes that will be present in the pdf i.e. $r=2$ will be bimodal, $r=3$ will have three modes. This complements the versatility of the ssvM pdf in accounting for potential multimodality in the data. The cdf of the ssvM is defined by [1]:

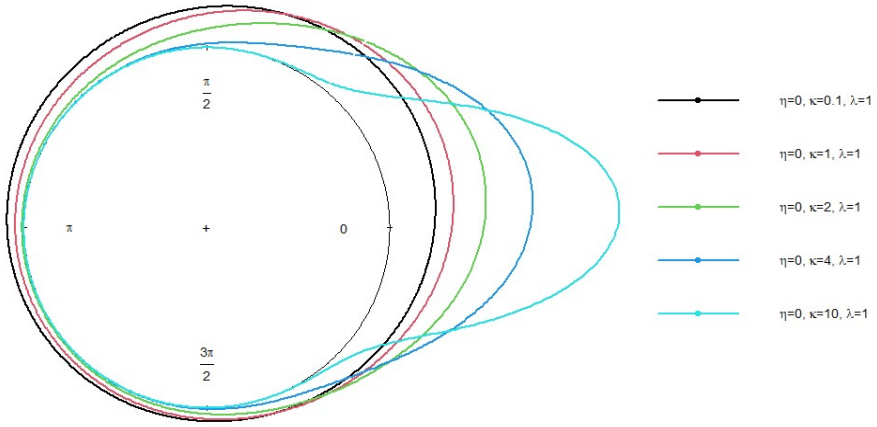
$$F(\theta)=F_0(\theta)+\frac{\lambda}{2\pi\kappa I_0(\kappa)}(\exp(-\kappa)-\exp(\kappa\cos r(\theta-\eta))),\quad -\pi\leq\theta<\pi,\ \ \kappa\geq 0,\ \ -1\leq\lambda\leq 1, \tag{15}$$

where $F_0(\theta)$ is the cdf of the base von Mises distribution.

Figure 3 illustrates pdf (14) for some arbitrary parameter choices:



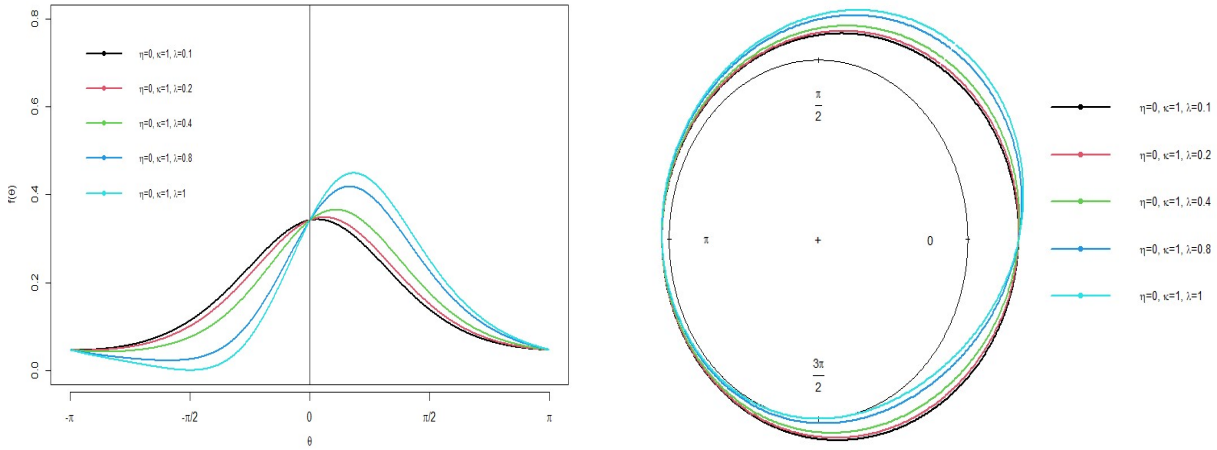
(a) The pdf of the ssvM



(b) The circular curve of the ssvM

Figure 3: The pdf and circular curve of the ssvM with $\lambda = 1$, $r = 1$ for $-\pi < \theta < \pi$ for different values of κ

As seen in Figure 3, as κ increases, the higher the peak of the pdf gets for a constant λ value. The lower the value of κ, the more the pdf is skewed to the left. Now, the behaviour when the value of λ changes is illustrated:

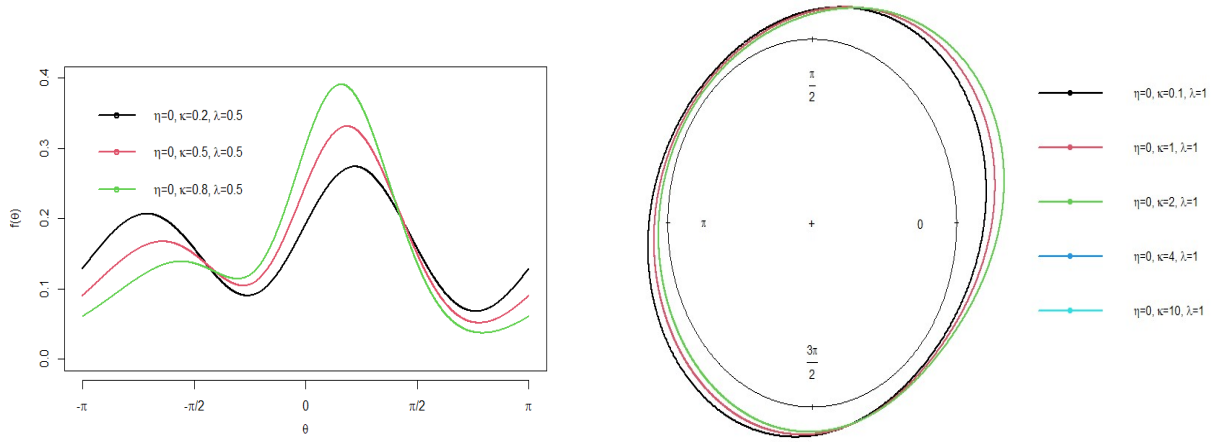


(a) The pdf of the ssvM

(b) The circular curve of the ssvM

Figure 4: The pdf and circular curve of the ssvM with $\kappa = 1$, $r = 1$ for $-\pi < \theta < \pi$ for different values of λ

As seen in Figure 4 above, as λ increases, the more the pdf skews to the left. Figure 5 below illustrates the bimodal nature of the (14):



(a) The pdf of the ssvM

(b) The circular curve of the ssvM

Figure 5: The pdf and circular curve of the ssvM with $\lambda = 0.5$, $r = 2$ for $-\pi < \theta < \pi$ for different values of κ

The value of $r = 2$ illustrates the bimodal nature of the ssvM pdf in contrast to the previous unimodal pdfs in Figures 3 and 4 when $r = 1$. The bimodal curves in 5 indicate that as κ increases, the peak of the pdf curves also increases.

As part of the model investigation, the trigonometric moments are derived as a key theoretical characteristic for this circular model.

Theorem 1. If a random variable θ has the pdf given by (14), then the p^{th} trigonometric moment defined by (87)

is given by [1]:

$$\varphi_p = \exp(ip\eta)A_p(\kappa)\left(1 + \frac{ip\lambda}{\kappa}\right) \tag{16}$$

where $A_p(\kappa) = \frac{I_p(\kappa)}{I_0(\kappa)}$ and $I_p(\kappa)$ is the modified Bessel function of order p with form (86) and $I_0(\kappa)$ is the modified Bessel function of the first kind of order 0.

Proof.

$$\begin{aligned}
\varphi_p &= \int_{-\pi}^{\pi} \frac{\exp(ip\theta)\exp(\kappa\cos(\theta-\eta))}{2\pi I_0(\kappa)}(1+\lambda\sin(\theta-\eta))d\theta \\
&= \exp(ip\eta)\left[\int_{-\pi}^{\pi} \frac{\exp(ip\theta)\exp(\kappa\cos(\theta))}{2\pi I_0(\kappa)}d\theta + \int_{-\pi}^{\pi} \lambda\frac{\exp(ip\theta)\exp(\kappa\cos(\theta))}{2\pi I_0(\kappa)}\sin(\theta)d\theta\right] \\
&= \exp(ip\eta)\left[\varphi_p^* + \lambda\int_{-\pi}^{\pi} \frac{\exp(ip\theta)\exp(\kappa\cos(\theta))}{2\pi I_0(\kappa)}\frac{i(\exp(-i\theta)-\exp(i\theta))}{2}d\theta\right]
\end{aligned}$$

where $\varphi_p^* = A_p(\kappa)$ is the p$^{\text{th}}$ trigonometric moment of the von Mises distribution

$$\begin{aligned}
&= \exp(ip\eta)\left[\varphi_p^* + \frac{i\lambda}{2}\left(\int_{-\pi}^{\pi} \frac{\exp(i\theta(p-1))\exp(\kappa\cos(\theta))}{2\pi I_0(\kappa)}d\theta - \int_{-\pi}^{\pi} \frac{\exp(i\theta(p+1))\exp(\kappa\cos(\theta))}{2\pi I_0(\kappa)}d\theta\right)\right] \\
&= \exp(ip\eta)\left[\varphi_p^* + \frac{i\lambda}{2}\left(\varphi_{p-1}^* - \varphi_{p+1}^*\right)\right] \\
&= \exp(ip\eta)\left[A_p(\kappa) + \frac{i\lambda}{2}\left(A_{p-1}(\kappa) - A_{p+1}(\kappa)\right)\right] \\
&= \exp(ip\eta)\left[A_p(\kappa) + \frac{ip\lambda}{\kappa}A_p(\kappa)\right]
\end{aligned}$$

which leaves the final result.

□

2.2 Distance

2.2.1 Power Lindley Distribution

A random variable T has the Lindley distribution if it has the pdf [14]:

$$\begin{aligned}
f(t) =& \frac{\beta^2}{\beta+1}(1+t)\exp(-\beta t), \quad t>0, \beta>0, \\
=& p\xi_1(t) + (1-p)\xi_2(t)
\end{aligned} \tag{17}$$

where $p = \frac{\beta}{\beta+1}$, $\xi_1(t) = \beta\exp(-\beta t)$ and $\xi_2(t) = \beta^2 t\exp(-\beta t)$. This illustrates the Lindley distribution is a mixture of an exponential distribution (88) (see Appendix) with scale parameter β and a Gamma distribution (3) with a shape parameter equal to 2 and scale parameter β with a mixing proportion of p as defined above. By applying the power transformation $X = T^{\frac{1}{\alpha}}$ to this variable, the pdf of the power Lindley is obtained as shown

below [14]:

$$\begin{aligned} f(x) =& \frac{\alpha\beta^2}{\beta+1}(1+x^\alpha)x^{\alpha-1}\exp(-\beta x^\alpha) \\ =& ps_1(x) + (1-p)s_2(x), \end{aligned} \tag{18}$$

where $p = \frac{\beta}{\beta+1}$, $s_1(x) = \alpha\beta x^{\alpha-1}\exp(-\beta x^\alpha)$ and $s_2(x) = \alpha\beta^2 x^{2\alpha-1}\exp(-\beta x^\alpha)$.

It is observed that this power Lindley distribution, denoted by $PL(\beta, \alpha)$ is also a mixture of a Weibull distribution (7) with shape parameter α and scale parameter β with a generalised Gamma distribution (89) (see Appendix) with shape parameters of α and 2, a scale parameter of β and with a mixing proportion of p (18).

The intrigue about $PL(\beta, \alpha)$ lies in its shape characteristics and its mixture distribution with the Gamma and Weibull distributions whilst remaining tractable as a contender for the modelling of distances in the context of animal movement modelling. The shape characteristics of $f(x)$ are summarised below at the points $x = 0$ and at $x \to \infty$ [14]:

$$f(0) = \begin{cases} \infty, & \text{if } \alpha < 1, \\ \frac{\beta^2}{\beta+1}, & \text{if } \alpha = 1, \\ 0, & \text{if } \alpha > 1, \end{cases} \tag{19}$$

$$f(\infty) = 0, \tag{20}$$

which indicates that the power Lindley distribution has three shapes for the pdf. These three shapes are illustrated on the next page by Figure 6 for arbitrary choices of β and α:

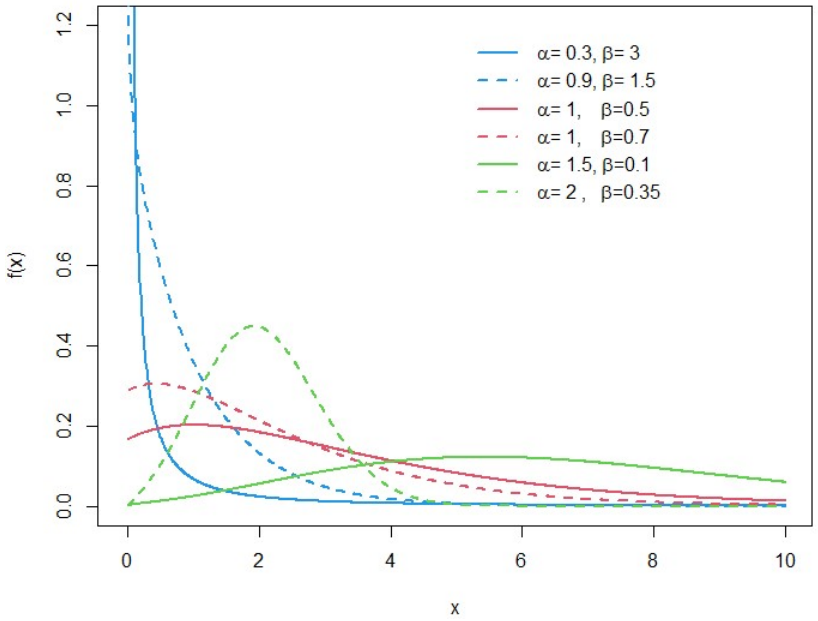


Figure 6: The pdf of $PL(\beta, \alpha)$ for various arbitrary parameter choices

The blue lines represent the first case, where $\alpha < 1$ with $f(0)$ clearly tending to ∞. The red lines represent the second case, where $\alpha = 1$ with the third case being represented by the green lines. These pdf lines indicate the versatility in pdf shape of $PL(\beta, \alpha)$, making this an interesting consideration in the context of animal movement. The n-th raw moment of $PL(\beta, \alpha)$ is given by [14]

$$\mu'_n = E(X^n) = \frac{n\Gamma(\frac{n}{\alpha})[\alpha(\beta+1)+n]}{\alpha^2\beta^{\frac{n}{\alpha}}(\beta+1)}. \tag{21}$$

Using equation (21) an expression for the mean and variance is given by:

$$E(X^1) = \mu = \frac{\Gamma(\frac{1}{\alpha})[\alpha(\beta+1)+1]}{\alpha^2\beta^{\frac{1}{\alpha}}(\beta+1)} \tag{22}$$

$$E(X^2) - E(X)^2 = \sigma^2 = \frac{2\Gamma(\frac{2}{\alpha})[\alpha(\beta+1)+2]\alpha^2(\beta+1) - \Gamma^2(\frac{1}{\alpha})[\alpha(\beta+1)+1]^2}{\alpha^4\beta^{\frac{2}{\alpha}}(\beta+1)^2} \tag{23}$$

The maximum likelihood estimators for the parameters of $PL(\beta, \alpha)$ were previously derived in literature and are given by [14]:

$$\hat{\beta}(\hat{\alpha}) = \frac{-\left(\sum_{i=1}^n x_i^{\hat{\alpha}} - n\right) + \sqrt{\left(\sum_{i=1}^n x_i^{\hat{\alpha}} - n\right)^2 + 8n\sum_{i=1}^n x_i^{\hat{\alpha}}}}{2\sum_{i=1}^n x_i^{\hat{\alpha}}} \tag{24}$$

where $\hat{\alpha}$ is the solution to the following non-linear equation:

$$G(\alpha) = \frac{n}{\alpha} + \sum_{i=1}^{n} \frac{x_i^{\alpha} \ln(x_i)}{1 + x_i^{\alpha}} + \sum_{i=1}^{n} \ln(x_i) - \hat{\beta}(\hat{\alpha}) \sum_{i=1}^{n} x_i^{\alpha} \ln(x_i) = 0 \tag{25}$$

2.2.2 Gumbel Distribution

The Gumbel distribution belongs to a class of continuous extreme value distributions [8]. This Gumbel distribution is a special case of the generalised extreme value type 1 distribution [22] and is unimodal with a pdf [8]

$$f(x) = \frac{1}{\sigma} \exp\left(-\frac{x-\mu}{\sigma}\right) \exp\left(-\exp\left(-\frac{x-\mu}{\sigma}\right)\right), \quad -\infty < x < \infty, \tag{26}$$

where $-\infty < \mu < \infty$ represents the location parameter and $\sigma > 0$ represents the scale parameter. If a random variable X has this pdf, it is denoted by $X \sim$Gumbel(μ, σ). The corresponding cdf for the Gumbel distribution is given by [8]

$$F(x) = \exp\left(-\exp\left(-\frac{x-\mu}{\sigma}\right)\right). \tag{27}$$

In the case where $\mu = 0$ and $\sigma = 1$, this distribution is the standard extreme value distribution [8]. The Gumbel distribution is typically used for modelling the maximum or minimum of a number of samples of various distributions. The pdf is skewed to the right which is illustrated by Figure 7 for arbitrary parameter values:

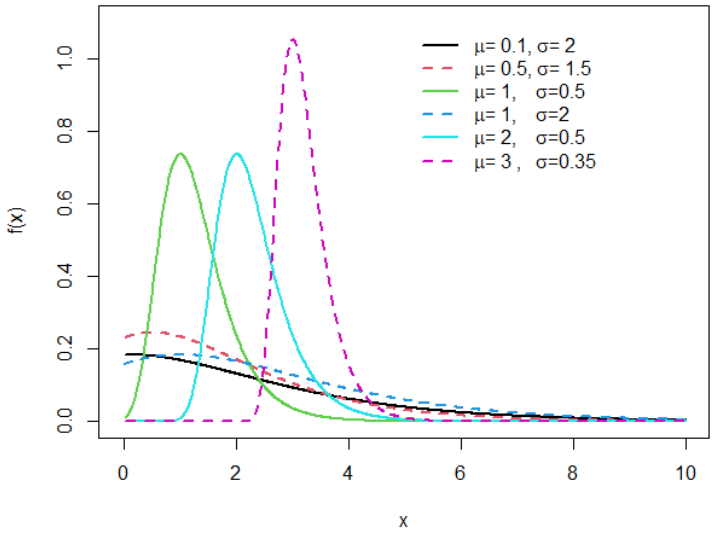


Figure 7: The pdf of the Gumbel(μ, σ) distribution for various parameter values

It is noted from the figure above that the graph will shift to the left as μ increases and the peak will be higher

as σ decreases. Although this distribution is not strictly positive, it is an interesting consideration for the modelling of the distance of animal movement on the condition that the values are strictly positive.

Theorem 2. The moment generating function (mgf) for the Gumbel(μ, σ) is given by [8]

$$M_X(t) = \Gamma(1 - \sigma t) \exp(\mu t). \tag{28}$$

Proof. The mgf of the Gumbel(μ, σ) distribution is:

$$\begin{aligned} M_X(t) =& E(\exp(tX)) \\ =& \int_{-\infty}^{\infty} \frac{1}{\sigma} \exp(tx) \exp\left(-\frac{x-\mu}{\sigma}\right) \exp\left(-\exp\left(-\frac{x-\mu}{\sigma}\right)\right) dx \end{aligned}$$

Let $Y = \exp\left(-\frac{x-\mu}{\sigma}\right)$ therefore $x = \mu - \sigma \ln(y)$

$$\begin{aligned} &= \int_{\infty}^{0} \exp\left(-t(\mu - \sigma \ln(y))\right) \exp(-y) dy \\ &= \exp(t\mu) \int_{0}^{\infty} y^{-\sigma t} \exp(-y) dy \\ &= \exp(t\mu)\Gamma(1 - \sigma t) \text{ since this is a gamma integral.} \end{aligned}$$

□

The derivation of the mean and variance of the Gumbel distribution is shown below.

Theorem 3. The mean and variance of the Gumbel(μ, σ) distribution is given by [8]:

$$E(X) = \mu + \sigma\gamma \tag{29}$$

$$Var(X) = \frac{\sigma^2 \pi}{6}, \tag{30}$$

where γ represents the Euler-Mascheroni constant (93) (see Appendix).

Proof. Using the mgf (28) to derive the expression for the mean yields the following:

$$\begin{aligned} E(X) =& \frac{\partial M_X(t)}{\partial t}\Big|_{t=0} \\ =& [\mu \exp(t\mu)\Gamma(1 - \sigma t) + \exp(t\mu)\Gamma'(1 - \sigma t)(-\sigma)]_{t=0} \\ =& \mu + \sigma\Gamma'(1) \\ =& \mu + \sigma\gamma \end{aligned}$$

as desired .

Firstly, in order to obtain the variance, $E(X^2)$ will need to be computed from the mgf (28) as shown below:

$$\begin{aligned}
E(X^2) =& \frac{\partial^2 M_X(t)}{\partial^2 t}\Big|_{t=0} \\
=& \mu[\mu\exp(t\mu)\Gamma(1-\sigma t) + \exp(t\mu)\Gamma'(1-\sigma t)(-\sigma)] - \sigma[\mu\exp(t\mu)\Gamma(1-\sigma t) + \exp(t\mu)\Gamma''(1-\sigma t)(-\sigma)]\Big|_{t=0} \\
=& \mu^2 - \mu\sigma\Gamma'(1) - \mu\sigma\Gamma'(1) + \sigma^2\Gamma''(1) \\
=& \mu^2 - 2\mu\sigma\Gamma'(1) + \sigma^2\Gamma''(1)
\end{aligned}$$

Thereafter, the variance can be obtained in the following manner,

$$\begin{aligned}
Var(X) =& E(X^2) - E(X)^2 \\
=& \mu^2 - 2\mu\sigma\Gamma'(1) + \sigma^2\Gamma''(1) - [\mu + \sigma\Gamma'(1)]^2 \\
=& \sigma^2\Gamma''(1) - \sigma^2\Gamma'(1)^2 \\
=& \sigma^2[\Gamma'(1)\Psi(1) + \Gamma(1)\Psi'(1) - \Gamma(1)^2\Psi^2(1)] \text{ using Result 9 in the Appendix} \\
=& \sigma^2\Psi'(1) \\
=& \frac{\sigma^2\pi^2}{6} \text{ since } \Psi'(1) = \frac{\pi^2}{6}.
\end{aligned}$$

□

The maximum likelihood estimators for the parameters of the Gumbel distribution which have been previously derived by Mahdi and Cenac are [27]:

$$\hat{\mu} = \sigma\left\{\ln(n) - \ln\sum_{i=1}^{n}\exp\left(-\left[\frac{x_i}{\sigma}\right]\right)\right\} \tag{31}$$

$$\bar{x} = \sigma + \frac{\sum_{i=1}^{n} x_i \exp\left(-\left[\frac{x_i}{\sigma}\right]\right)}{\sum_{i=1}^{n}\exp\left(-\left[\frac{x_i}{\sigma}\right]\right)} \tag{32}$$

where the estimate of σ is obtained explicitly from (32) and the estimate of μ is implicitly obtained from (31) after the estimation of σ.

3 Modelling Approach and Implementation

3.1 EM Algorithm

The Expectation Maximisation (EM) algorithm is an approach to performing maximum likelihood estimation in the presence of latent variables and missing values. This algorithm has a crucial role in the modelling performed in this section and is introduced. Suppose that there is a pdf function $f(\mathbf{x}, \mathbf{\Theta})$ which is governed by parameters set $\mathbf{\Theta}$ and a dataset of size K drawn from $f(\mathbf{x}, \Theta)$ [5]. Then, let $\boldsymbol{X} = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$ and make the assumption that the data vectors are independent and identically distributed, with the pdf being $f(\cdot)$. The likelihood of $\mathbf{\Theta}$ given the data is then

$$f(\mathbf{X}|\mathbf{\Theta}) = \prod_{i=1}^{K} f(\mathbf{x}_i|\mathbf{\Theta}) = L(\mathbf{\Theta}|\mathbf{X}). \tag{33}$$

The goal of maximum likelihood estimation is to find the Θ that maximises L, which is $\mathbf{\Theta}^*$ where

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\text{argmax}} L(\mathbf{\Theta}|\mathbf{X}) \tag{34}$$

When the optimisation of the likelihood function is mathematically intractable, the EM algorithm is typically used. As above, $\mathbf{X}$ is the observed data generated by $f(\cdot)$ and is called the incomplete data [5]. With the assumption that a complete dataset exists, say $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ where $\mathbf{Y}$ denotes the missing information which is unknown, random and presumed to be governed by an underlying distribution. Therefore a joint pdf is specified [5]:

$$\begin{aligned} f(\mathbf{z}|\mathbf{\Theta}) =& \frac{f(\mathbf{x}, \mathbf{y}, \mathbf{\Theta})}{f(\mathbf{\Theta})} \\ =& \frac{f(\mathbf{x}, \mathbf{y}, \mathbf{\Theta})}{f(\mathbf{\Theta})} * \frac{f(\mathbf{x}, \mathbf{\Theta})}{f(\mathbf{x}, \mathbf{\Theta})} \\ =& \frac{f(\mathbf{x}, \mathbf{y}, \mathbf{\Theta})}{f(\mathbf{x}, \mathbf{\Theta})} * \frac{f(\mathbf{x}, \mathbf{\Theta})}{f(\mathbf{\Theta})} \\ =& f(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) f(\mathbf{x}|\mathbf{\Theta}). \end{aligned} \tag{35}$$

Using the joint pdf (35), a new likelihood function can be defined called the complete likelihood:

$$L(\mathbf{\Theta}|\mathbf{Z}) = L(\mathbf{\Theta}|\mathbf{X}, \mathbf{Y}) = f(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta}). \tag{36}$$

This function is a random variable since $\mathbf{Y}$, the missing information is unknown, random and presumed to be governed by an underlying distribution. The original likelihood $L(\mathbf{\Theta}|\mathbf{X})$ is called the incomplete-data likelihood

[5].

The EM algorithm firstly finds the expected value of the complete-data log-likelihood $\log f(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta})$ with respect to the unknown data $\mathbf{Y}$ given the current parameter estimates and observed data $\mathbf{X}$. Note that no prior knowledge of the parameter is included, and it is rather assumed that the choice of the parameter vector is equally likely [6]. This expectation is defined as [5]:

$$\begin{aligned} Q(\mathbf{\Theta}, \mathbf{\Theta}^{i-1}) =& E\{\log f(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta})|\mathbf{X}, \mathbf{\Theta}^{(i-1)})\} \\ =& \int_{\mathbf{y}\in\gamma} \log f(\mathbf{X}, \mathbf{y}|\mathbf{\Theta}) f(\mathbf{y}|\mathbf{X}, \mathbf{\Theta}^{(i-1)}) d\mathbf{y} \end{aligned} \tag{37}$$

where $\mathbf{\Theta}^{(i-1)}$ represent the current parameter estimates being used to evaluate the expected value and $\mathbf{\Theta}$ are the new parameters being optimised to increase Q. Since $\mathbf{X}$ and $\mathbf{\Theta}^{(i-1)}$ are constants and $\mathbf{Y}$ is the random variable which is governed by the distribution $f(\mathbf{y}|\mathbf{X}, \mathbf{\Theta}^{(i-1)})$, the expectation may be rewritten into integral form shown in (37) where γ denotes the space of values which $\mathbf{y}$ may assume.

The second step of the EM algorithm is the maximisation of the expectation in (37). The following is determined:

$$\mathbf{\Theta}^{(i)} = \underset{\mathbf{\Theta}}{\text{argmax}}\ Q(\mathbf{\Theta}|\mathbf{\Theta}^{(i-1)}).$$

Convergence of the algorithm is guaranteed since the EM algorithm positively converges to a local maximum of the likelihood function. The two steps are repeated until two successive estimates are equal or approximately equal within an arbitrary threshold $\epsilon > 0$, $\mathbf{\Theta}^* = \mathbf{\Theta}^{(i)} = \mathbf{\Theta}^{(i-1)}$ at some i^{th} iteration [30].

The EM algorithm does however have some disadvantages. If there are many latent parameters in the model, the number of local peaks tend to increase. The algorithm converges towards a local maximum and not the global maximum, which is often difficult to determine.

3.2 Proposed Model

Using the work of Nicosia et al. [32] as a framework, alternative considerations for the directional and distance components of the BCRW model are proposed. For the directional component, the ssvM (14) is proposed. For the distance component, the power Lindley (18) and Gumbel distribution (26) are considered within this context.

3.2.1 Hidden State Model Outline And Assumptions

Consider a data set which consists of the time series:

$$\{y_t, d_t, \mathbf{x_t}, \mathbf{z_t}, t = 0, \ldots, T\} \tag{38}$$

where

- $y_t \in [0, 2\pi)$ represents the direction between the animal's location from time step t to time step $t+1$
- $d_t \geq 0$ represents the distance between the animal's location from time step t to time step $t+1$
- $\mathbf{x_t} = (x_{1t}, \ldots, x_{nt})$ are the values of the n explanatory angular (directional) variables which are measured which could potentially be used to predict y_t or d_t.
- $\mathbf{z_t} = (z_{1t}, \ldots, z_{nt})$ are the values of the n explanatory real variables which are measured which could potentially be used to predict y_t or d_t.

The explanatory variables x_{jt} and z_{jt}, $j = 1, \ldots, n$ are associated to the direction and distances of the animal being observed with respect to the position of the animal at the previous time step i.e. $t-1$. It should be noted that the explanatory variable z_{nt} may also be an indicator variable in certain circumstances. The set of all observed distances and directions are denoted by $(y_{0:T}, d_{0:T}) = \{(y_t, d_t), t = 0, \ldots, T\}$ for simplicity. The observed information for the variables involved, i.e. directions, distances and explanatory variables are denoted by a filtration $\mathscr{F}_t^0$. The reason for this denotation is to differentiate the observed data from unobserved data [32].

The hidden Markov process S_t (see Appendix A (5.2.6)) , $t = 1, \ldots, T$ is used in this model to represent the behaviour i.e. state in which the animal is in at time t. This is due to the fact that animals exhibit multiple behavioural patterns over time. These changes in behavioural patterns are not observed over time, so this hidden state will account for these behavioural changes over the various time steps. The set of the possible hidden states are denoted by $S_{0:T} = \{S_0, \ldots, S_T\}$. The complete data filtration is denoted by $\mathscr{F}_t^c$ which is the filtration generated by $\mathscr{F}_t^0$ and the hidden information until time t.

The joint pdf for the complete data is therefore:

$$f(y_{0:T}, d_{0:T}, S_{0:T}) = \prod_{t=1}^{T} g(S_t | \mathscr{F}_{t-1}^c) k(y_t, d_t | S_t, \mathscr{F}_{t-1}^c) \tag{39}$$

where $g(\cdot)$ represents the pdf of the hidden data and $k(\cdot)$ represents the pdf of the observed data.

3.2.2 Directional Random Walk Model

In order for the directional-distance proposals to remain theoretically and empirically practical, specific assumptions on the joint pdf in (39) need to be made which are highlighted below:

1. The observed processes $y_{0:T}$ and $d_{0:T}$ i.e. direction and distance respectively are independent given the hidden process $S_{0:T}$. This is expressed in the following manner

$$k(y_t, d_t|S_t, \mathscr{F}_{t-1}^c) = f(y_t|S_t, \mathscr{F}_{t-1}^c)h(d_t|S_t, \mathscr{F}_{t-1}^c), \quad t = 1, \ldots, T. \tag{40}$$

   Furthermore, it is assumed that the processes $y_{0:T}$ and $d_{0:T}$ are of Markovian order 1 (see Appendix A (5.2.4)) with respect to the hidden process S_t. This is assumed for both simplicity and computational ease. Note that $h(d_t|S_t, \mathscr{F}_{t-1}^c)$ represents the conditional pdf of the distance d_t.

2. Given the hidden process $S_{0:T}$, it is assumed

$$f(y_t|S_t, \mathscr{F}_{t-1}^c) = f(y_t|S_t, \mathscr{F}_{t-1}^o), \quad t = 1, \ldots, T. \tag{41}$$

   Given that y_t is Markovian of order 1 the distribution of y_t does not depend on the previous states of the hidden process and only depends on the current hidden state S_t. This relates to the Markov property mentioned in Appendix A (5.2.4). It should be noted that $f(y_t|S_t, \mathscr{F}_{t-1}^c)$ depends on S_t and $\{y_{t-r}\}_{r<t}$ as well as environmental variables which were observed in $\mathscr{F}_{t-1}^o$. Therefore, it is supposed that the pdf of y_t given the observed information contained information in $\mathscr{F}_{t-1}^o$ with the knowledge that the hidden process is in state S at time t is denoted by $f_s(\cdot|\mathscr{F}_{t-1}^o)$.

3. Given the hidden process $S_{0:T}$, it is assumed that

$$h(d_t|S_t, \mathscr{F}_{t-1}^c) = h(d_t|S_t), \quad t = 1, \ldots, T. \tag{42}$$

   This implies that the distance is independent on the previous distances taken in the previous time steps. i.e. d_t is independent of $d_{t-r}, r = 1, \ldots, t$ for time steps $t = 1, \ldots, T$. This assumption is also made for computational ease and modelling simplicity. The consequence of not making this assumption would be to model the distances auto-regressively which would heavily impact computational time.

Specific parametric forms are considered for both $f_s(\cdot)$ and $h_s(\cdot)$ for the purpose of this study in order to provide different parametric considerations for direction and distance. For the case of the distance $h_s(\cdot)$, any positive real line pdf may be considered. The power Lindley and Gumbel distributions are being considered for

$h_s(\cdot)$ in order to compare their respective modelling performances with the Gamma and Weibull distributions [32], which are popularly used in this context for animal movement modelling.

3.2.3 Circular Multivariate Regression Model

In this section, the circular regression models for the BCRW are discussed. The BCRW model theory has been introduced in (5.2.3) together with the linear-circular regression theory in Section (5.1) in Appendix A. Nicosia et al. [32] considered y_t to be von Mises distributed with its mean direction being dependent on y_{t-1} together with other explanatory variables and a homogenous error [32]. This homogenous error depended on a fixed concentration parameter κ. In these findings, it was discovered that multimodality often plagued the estimation procedure, and highlighted that the log-likelihood for estimation was multimodal [32]. To combat this issue, a consensus error model was adopted which was defined in a previous paper by Nicosia et al. [31]. This method is adapted to the circular model being considered for comparison, namely the ssvM distribution [1]. The theory of these distributions have been discussed in Section 2.

A consensus error model for the direction y_t with knowledge of the animal being in state k depends on the vector [32]

$$V_t^k = \kappa_0^{(k)} \begin{pmatrix} \cos(y_{t-1}) \\ \sin(y_{t-1}) \end{pmatrix} + \sum_{i=1}^{n} \kappa_i^{(k)} z_{it} \begin{pmatrix} \cos(x_{it}) \\ \sin(x_{it}) \end{pmatrix}, \quad t = 1, \ldots, T, \tag{43}$$

where $\kappa^{(k)} = (\kappa_0^{(k)}, \ldots, \kappa_n^{(k)})$ represent the unknown parameters dependent on state k. In terms of the von Mises distribution, its mean direction denoted by $\mu_t^{(k)}$ is the direction of $V_t^{(k)}$. The parameters $\kappa_i^{(k)}$ quantify how target i influences the animal's directional movement. The same consensus model is adapted for the ssvM.

The concentration parameter of the von Mises distribution denoted by $\ell_t^{(k)}$ is the length of the vector (43). This indicates that the concentration parameter $\kappa_i^{(k)}$ is dependent on the relationship between the directional targets under consideration. The concentration parameter $\ell_t^{(k)}$ has a large value if y_{t-1} and the targets all point in the same direction and therefore, the distribution of y_t is concentrated around the mean direction. Using these as underlying assumptions for the model, we have that the pdf of the direction given the observed data and concentration parameter is [32]

$$f_k(y_t|\mathscr{F}_{t-1}^c; \kappa^{(k)}) = \frac{\exp\{\ell_t^{(k)} \cos(y_t - \mu_t^{(k)})\}}{2\pi I_0(\ell_t^{(k)})}, \quad , t = 1, \ldots, T. \tag{44}$$

Since $\mu_t^{(k)}$ and $\ell_t^{(k)}$ are the direction and length of the vectors (43) and (44), the pdf then becomes [32]

$$f_k(y_t|\mathscr{F}_{t-1}^c; \kappa^{(k)}) = \frac{\exp\{\kappa_0^{(k)} \cos(y_t - y_{t-1}) + \sum_{i=1}^{n} \kappa_i^{(k)} z_{it} \cos(y_t - x_{it})\}}{2\pi I_0(\ell_t^{(k)})}, \quad , t = 1, \ldots, T. \tag{45}$$

This parametrisation is of importance since it results in a numerically stable model with $\kappa_i^{(k)}$ being the canonical parameters of a distribution which belongs to the exponential family. Similarly, for the ssvM, the pdf of the direction is as follows:

$$f_k(y_t|\mathscr{F}_{t-1}^c;\kappa^{(k)}) = \frac{\exp\{\kappa_0^{(k)}\cos(y_t - y_{t-1}) + \sum_{i=1}^n \kappa_i^{(k)} z_{it}\cos(y_t - x_{it})\}}{2\pi I_0(\ell_t^{(k)})} \times \left(1 + \lambda\left[\sin(y_t - y_{t-1}) + \sum_{i=1}^n z_{it}\sin(y_t - x_{it})\right]\right), \quad t = 1,\ldots,T, -1 \leq \lambda \leq 1. \tag{46}$$

Many special cases of this model exists in literature. These special cases extend to the consensus model construction, number of hidden states, model assumptions, random walks and multi-state frameworks. These special cases fall outside the scope of the current study.

3.2.4 The Markov Processes For The Hidden States

The hidden process $S_{0:T}$ is a homogenous Markov chain. A Markov chain is homogenous if and only if the transition probabilities are not dependent of time t. The theory for the Markov chains and hidden Markov processes are introduced in within this context, which implies that at any time step t, the animal is in a state ranging from $\{1,\ldots,K\}$ and $g(S_t|\mathscr{F}_{t-1}^c) = g(S_t|S_{t-1})$ [32]. This allows the description of $S_{0:T}$ as a discrete-time homogenous multinomial process. This permits the definition of the sequence $\{\mathbf{S}_t, t = 0,\ldots,T\}$ of multinomial vectors $\mathbf{S}_t = (S_{1t},\ldots,S_{Kt})$ where $S_{it} = 1$ and $S_{i't} = 0$ for all $i' \neq i$ whenever $S_t = i$ for $i, i' = 1,\ldots,K$.

At the time $t = 0$, it is set that $P(S_{k0} = 1) = (\pi_0)_k$ such that $(\pi_0)_k \geq 0$, $k = 1,\ldots,K$ and $\sum_{k=1}^K (\pi_0)_k = 1$. It should be noted that $(\pi_0)_k$ represents the hidden process' initial distribution. The assumption is made that this initial distribution of the hidden process $\{(\pi_0)_k \geq 0, k = 1,\ldots K\}$ is known. The transition probabilities are denoted by $\pi_{lk} = P(S_{kt} = 1|S_{l,t-1} = 1)$ for $l, k = 1, 2,\ldots,K$. The hidden process contributes to the complete data pdf given in (39) as a function of $S_{0:T}$ and the transition probabilities in the following way [32]

$$\prod_{t=1}^T g(S_t|\mathscr{F}_{t-1}^c) = \prod_{t=1}^T \prod_{l=1}^K \prod_{k=1}^K \pi_{lk}^{S_{l,t-1}S_{kt}}. \tag{47}$$

The aforementioned methodology holds for any number of states K, but the case of $K = 2$ is the only case considered in this scenario.

3.2.5 Inferential procedures and EM algorithm

In this section, the inferential procedures and EM algorithm are explained. A procedure to estimate $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\kappa}, \boldsymbol{\phi})$, where

- $\boldsymbol{\pi}$ represents the transition probabilities in a $K \times (K-1)$ vector,
- $\boldsymbol{\kappa}$ represents the $K \times (p+1)$ unknown parameters for the directional model and
- $\boldsymbol{\phi}$ represents the $2K$ parameters of the distance variable model.

Given that we have the series of observations as before denoted by (38), the likelihood function of the parameters in the model are written as a product of the one-step ahead predictive pdfs,

$$L(\theta) = \prod_{t=1}^{T} \left\{ \sum_{k=1}^{K} f_k(y_t|\mathscr{F}_{t-1}^0, \kappa^{(k)}) h_k(d_t, \phi^{(k)}) P(S_{kt} = 1|\mathscr{F}_{t-1}^0, \theta) \right\} \tag{48}$$

where $P(S_{tk} = 1|\mathscr{F}_{t-1}^0, \theta)$ for $t = 1, \ldots, T$ represents the predictive probabilities which are recursively calculated using a filtering-smoothing algorithm which will be presented in the next sub-section permitting the evaluation of the likelihood function (48). The EM algorithm is used to estimate the likelihood to address modelling shortcomings by previous authors such as [24]. The EM algorithm permits the prediction of the underlying state of the animal at each time point which leads to greater knowledge of their movements and behavioural patterns as well as other fascinating ecological significances.

The primary function of the EM algorithm (see section 3.1) is to maximise the likelihood in the presence of missing or unobserved data, which is the current case. The observed data consists of the direction $y_{0:T}$, the distance $d_{0:T}$ and the unobserved data are the hidden states $S_{0:T}$. To evaluate the EM algorithm, the complete data log-likelihood function should be computed which follows from (39) and (48):

$$\begin{aligned} \log L_{\text{complete}}(\theta; \mathscr{F}_{t-1}^0) = &\sum_{t=1}^{T} \sum_{r=1}^{K} \sum_{k=1}^{K} S_{r,t-1} S_{kt} \log(\pi_{rk}(n_r, q_r)) + \sum_{t=1}^{T} \sum_{k=1}^{K} S_{kt} \log(f_k(y_t|\mathscr{F}_{t-1}^0, \kappa^{(k)})) \\ &+ \sum_{t=1}^{T} \sum_{k=1}^{K} S_{kt} \log(h_k(d_t|\phi^{(k)}) \end{aligned} \tag{49}$$

Let $\hat{\theta}_n$ denote the estimate of θ as previously defined after the n^{th} iteration of the EM algorithm. The $(n+1)^{\text{th}}$ iteration begins with one of the iteration of the E-step which computes the expected value of the complete data log-likelihood with regards to the missing data's conditional distribution given the observed data which is expressed as follows:

$$Q(\theta|\hat{\theta}_n) = E_{S_{0:T}}[\log L_{\text{complete}}(\theta; \mathscr{F}_T^c)|\mathscr{F}_T^0, \hat{\theta}_n]$$

$$= \sum_{t=1}^{T}\sum_{r=1}^{K}\sum_{k=1}^{K} E(S_{r,t-1}S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)\log(\pi_{rk}(n_r, q_r)) + \sum_{t=1}^{T}\sum_{k=1}^{K} E(S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)\log(f_k(y_t|\mathscr{F}_{t-1}^0, \kappa^{(k)}))$$
$$+ \sum_{t=1}^{T}\sum_{k=1}^{K} E(S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)\log(h_k(d_t|\phi^{(k)})) \tag{50}$$

Hereafter, the value of $\hat{\theta}_{n+1}$ is calculated in the M-step as the value of θ which will maximise $Q(\theta|\hat{\theta}_n)$.

The E-Step

Within (50), there are two expected values which will have to be computed, namely $E(S_{r,t-1}S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)$ and $E(S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)$. The computation of these expectations is done using a filtering smoothing algorithm specifically for Markov chains which will start from $t = 0$. This algorithm will compute the "filtering" probabilities denoted by $P(S_T|\mathscr{F}_t^0)$ with the use of predictive probabilities $P(S_t|\mathscr{F}_{t-1}^0)$ [32]. Finally, the final filtering probability denoted by $P(S_T|\mathscr{F}_T^0)$ is the probability which is used to compute the smoothing probabilities $P(S_t|\mathscr{F}_T^0)$ using Bayes theorem. The filtering smoothing algorithm is further detailed below.

Filtering Smoothing Algorithm

The filtering smoothing algorithm for this Markov specification is presented. This algorithm will compute the two expectations in (50) which both involve the hidden state $S_{k,t}$ which is unobserved and conditional on the observed data $\mathscr{F}_T^0$ [32]. Firstly, the expectations are expressed as probabilities:

$$E(S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n) = P(S_{kt} = 1|\mathscr{F}_T^0, \hat{\theta}_n) \tag{51}$$
$$E(S_{r,t-1}S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n) = P(S_{r,t-1} = 1|S_{kt} = 1, \mathscr{F}_T^0, \hat{\theta}_n)P(S_{kt} = 1|\mathscr{F}_T^0, \hat{\theta}_n) \text{ by conditional probability} \tag{52}$$

Probability (52) is then computed using Bayes theorem due to the fact that S_{t-1} is independent of the observed data for $t = 1, \ldots, T$ given S_t and $\mathscr{F}_{t-1}^0$. This therefore satisfies the requirements of Bayes theorem yielding the following:

$$P(S_{r,t-1} = 1|S_{kt} = 1, \mathscr{F}_T^0, \hat{\theta}_n) = \frac{\hat{\pi}_{rk}^{(n)} P(S_{r,t-1} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n)}{\sum_{i=1}^{K} \hat{\pi}_{ik}^{(n)} P(S_{i,t-1} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n)}, \quad k = 1, \ldots, K, t = 0, \ldots, T. \tag{53}$$

Thereafter, the posterior expectations (51) and (52) need to be computed using the filtering smoothing algorithm for the E-step of the EM algorithm for the $(n+1)^{\text{th}}$ step of the EM algorithm.

Filtering The probability $P(S_{mt} = 1|\mathscr{F}_T^0, \hat{\theta}_n)$ will be computed for every $m = 1, \dots T$ in the following manner [13]:

$$P(S_{mt} = 1|\mathscr{F}_T^0, \hat{\theta}_n) = \frac{f_m(y_t|\mathscr{F}_{t-1}^0, \hat{\theta}_n) h_m(d_t, \hat{\theta}_n) P(S_{mt} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n)}{\sum_{k=1}^K f_k(y_t|\mathscr{F}_{t-1}^0, \hat{\theta}_n) h_k(d_t, \hat{\theta}_n) P(S_{kt} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n)}, \tag{54}$$

where

$$P(S_{m1} = 1|\mathscr{F}_0^0, \hat{\theta}_n) = \sum_{k=1}^K \hat{\pi}_{km}^{(n)} (\pi_0)_k \tag{55}$$

$$P(S_{mt} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n) = \sum_{k=1}^K \hat{\pi}_{km}^{(n)} P(S_{k,t-1} = 1|\mathscr{F}_{t-1}^0, \hat{\theta}_n) \quad \text{for } t = 2, \dots, T \tag{56}$$

Smoothing The probability $P(S_{mt} = 1|\mathscr{F}_T^0, \hat{\theta}_n)$ will be computed for every $m = 1, \dots T$ in the following manner [13]:

Step 1: For $t = T$, set $P(S_{mT} = 1|\mathscr{F}_T^0, \hat{\theta}_n)$ which is the conditional probability that is computed at the last filtering step (54).

Step 2: Recursion step: For $t = T - 1, \dots, 0$, the following is computed [32]:

$$P(S_{mT} = 1|\mathscr{F}_T^0, \hat{\theta}_n) = \sum_{k=1}^K \frac{\hat{\pi}_{mk}^{(n)} P(S_{mt} = 1|\mathscr{F}_t^0, \hat{\theta}_n) P(S_{m,t+1} = 1|\mathscr{F}_T^0, \hat{\theta}_n)}{\sum_{i=1}^K \hat{\pi}_{ik}^{(n)} P(S_{it} = 1|\mathscr{F}_t^0, \hat{\theta}_n)}. \tag{57}$$

This filtering smoothing algorithm highlighted above enables the evaluation of the predictive probabilities in a recursive manner using (56) resulting in the computation of the likelihood (48). Below, the M-step and further details regarding the EM algorithm are outlined below.

The M-step

Within (50), the individual elements composing the sum are maximised separately due to the fact that they each depend on different parameter sets. In the case where the hidden states are of a Markovian nature, the maximisation of the hidden process is expressed by the expected number of transitions made from state r to state k divided by the number of transitions leaving state r [32]:

$$\hat{\pi}_{rk}^{(n+1)} = \frac{\sum_{t=1}^T E(S_{r,t-1} S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)}{\sum_{t=1}^T E(S_{r,t-1}|\mathscr{F}_T^0, \hat{\theta}_n)}, \quad r, k = 1, \dots, K. \tag{58}$$

It should be noted that the log-likelihood for the directional component of the model is concave, due to the nature of the von Mises and ssvM distributions which allows the maximum to be calculated with ease. The M-step of the EM algorithm depends on $h_k(d_t|\phi^{(k)})$ which represents the distance component of the model. The estimates

for this pdf is calculated using numerical optimisation such as the Newton-Raphson algorithm of the weighted log-likelihood.

3.2.6 Sampling distributions

When computing the EM algorithm, there are certain quantities which are crucial to the inference of the model which are not directly computed by the algorithm such as the numerical value of the maximised log-likelihood for the observed data. The filtering smoothing algorithm which was detailed above may be used to evaluate this likelihood (48). The filtering smoothing algorithm further allows the evaluation of the probability that the animal is in a state k by using the value of $E(S_{kt}|\mathscr{F}_T^0, \hat{\theta}_n)$ which is in the "smoothing" step of the algorithm specifically by (57). The computation of the maximised log-likelihood enables the numerical approximation of the negative Hessian matrix. The Hessian matrix is a square matrix of the second derivatives of the parameters of the log-likelihood. The inverse of this Hessian matrix is denoted by ν which is the estimate of the variance matrix of the maximum likelihood estimators [32]. In this study, the `R` function `optim` is used to compute the numerical approximation of the Hessian matrix.

Due to the mathematically complex nature of the model, the EM algorithm may converge to spurious or local maxima of the likelihood function (48). This is typically caused by the likelihood being unbounded which is a phenomenon commonly found in mixture models [32]. To combat this issue and maintain consistency in the algorithm, the EM algorithm is run using many random initial starting values for a few iterations and perform a check for spurious and local maxima. Thereafter, parameter values are chosen based on which values yield the highest likelihood values for use as the starting values of the EM algorithm which is run until convergence [4]. The combination of short and long-run iterations of the EM algorithm forms a good strategy for the avoidance of local and spurious maxima. The quasi-Newton algorithm is then run with a single iteration in order to obtain an estimate for inverse of the Hessian matrix of the observed log-likelihood which is estimated at the maximum likelihood estimates which are obtained from the EM algorithm. The quasi-Newton algorithm simply indicates that the log-likelihood and its respective gradient are calculated recursively. The algorithm of finding the global maximum of the observed log-likelihood function is described below [32]:

Initial Steps

- Let $\theta_1, \ldots, \theta_N$ represent N random starting values for the EM algorithm. In this study, $N = 50$ was selected.
- For $i = 1, \ldots N$, the EM algorithm will run until:

    1. The first 50 iterations or

2. The largest relative difference in parameter values between consecutive iterations is less than 0.01. The final estimators obtained at the end are denoted by $\hat{\theta}_i, i = 1, \ldots, K$.

The avoidance of spurious maxima

- For each of the $\hat{\theta}_i$, the stationary distribution of the Markov chain is denoted by $\hat{\nu}_k^{(i)}, k = 1, \ldots, K, i = 1, \ldots, N$.
- Only keep the $\hat{\theta}_i$ such that the following condition holds:

$$\min_{k=1,\ldots,K} \hat{\nu}_k^{(i)} > \epsilon \text{ and } \max_{r=1,\ldots,p;k=1,\ldots,K} |\kappa_r^{(k)(i)}| < M.$$

It should be noted that in this study, $M = 100$ and $\epsilon = 0.001$ was set for computational ease.

The avoidance of local maxima

- The initial value was set as $\theta_0 = \text{argmax}\, L(\hat{\theta}_i)$. This then avoids the local maxima.

The Long-run EM algorithm

- Commence the EM algorithm using the θ_0 as set in the previous step and run the algorithm until the first of
  1. The largest relative difference in the parameter values between successive iterations is less than 10^{-8} or
  2. 10 000 iterations.

  This long-run algorithm is conservative with the number of iterations to allow the algorithm sufficient time to converge to a final solution with respect to all parameters.

The Quasi-Newton iteration

- A single iteration of the quasi-Newton algorithm is run using the output of the long-run EM algorithm as the starting values in order to obtain the final global maximum likelihood estimators of parameters in the model as well as the estimation of their respective variance matrix i.e. the Hessian matrix.

It should be noted that the initial distribution $(\pi_0)_k, k = 1, \ldots, K$ is involved in the calculation of the complete or observed data likelihood, as shown above in the algorithm [32]. Due to this involvement, the model is fitted twice. Firstly, $(\pi_0)_k = \frac{1}{K}$ which is the uniform distribution across the K states. Secondly, the model is refitted using $(\pi_0)_k = \hat{\nu}_k$ which is the stationary distribution of the Markov chain computed using the first model fit. This allows the model to have greater accuracy and more efficient convergence.

3.2.7 Model Selection

The determination of the number of hidden states present in the model is typically an arduous task and is unknown. In this scenario, two hidden hidden states are assumed based on previous literature and studies performed by various authors in this research field [32]. The selection of two states for the hidden process enables interpretability within the animal movement model environment. The directional targets and environmental features to be considered were selected using the AIC, BIC and Wald's test criteria. These criteria together with the log-likelihood values were used to determine which combination of alternative direction and distance models could be useful when modelling animal movement.

3.3 Results

In this section, the results for the fitted BCRW model (37) and the respective alternate parametric considerations are presented. A variety of `R` functions are used whilst fitting the model which is available in Appendix B within the code. This model was applied to data about the movement of forest caribou in Canada. This data involved 23 animals with various home ranges. Locations were measured every 4 hours during the 2006-2007 winter period, a total number of 617 observations. The data consists of four variables, namely:

- y_t: the direction of the animal at time point t.
- x_{cut}: the direction to the closest cut (which is a forest stand which has been cut within a period ranging from 5 to 20 years ago).
- x_{center}: the direction pointing towards the centroid of a cluster of recently visited locations.
- d_t: the distance between the animal's location from time step t to time step $t+1$ in kilometres.

Before the modelling results are presented, the descriptive statistics of the data are shown for the reader to understand the nature of the data and its variables. These descriptive results are shown in Table 1:

	Value
Coefficient of Skewness	0.1400
Kurtosis	-2.5076
Variance	0.0005
Mean	0.0014

Table 1: The descriptive statistics for direction y_t

Based on Table 1, y_t is slightly skewed to the right based on the coefficient of skewness and has a flat distribution with thin tails based on the kurtosis value. The mean value is small, indicating that on average, the direction of the animal at time point t will not change by a large angle. The low variance could indicate that

the data consists of smaller turning angles majority of the time for the animals at the time points observed. The corresponding rose plot for direction y_t is shown in Figure 8:

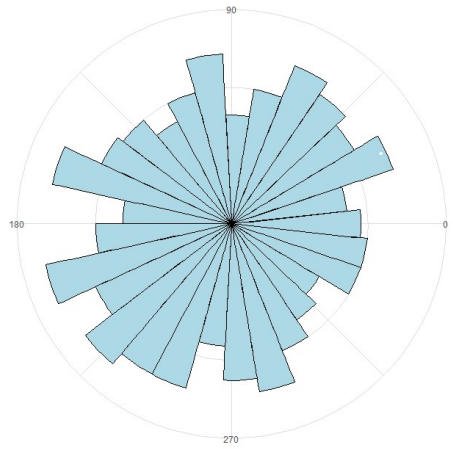


Figure 8: Rose plot for direction y_t

The rose plot represented by Figure 8 indicates that the animal paths tend to concentrate around direction zero and seems to be quite uniform. This would imply that the animal would turn around and does not necessarily have a preferred direction. The descriptive results for the distance d_t are shown in Table 2:

	Values
Min	0.0015
First Quartile	0.0642
Median	0.1259
Mean	0.3120
Third Quartile	0.2490
Max	10.0381

Table 2: The descriptive statistics for distance d_t

It is noted that majority of the values of d_t lie between 0 and 1, indicating that the animals tended to travel small distances within the four hour intervals. A map indicating the animal trajectory is presented in Figure 9 [32]:

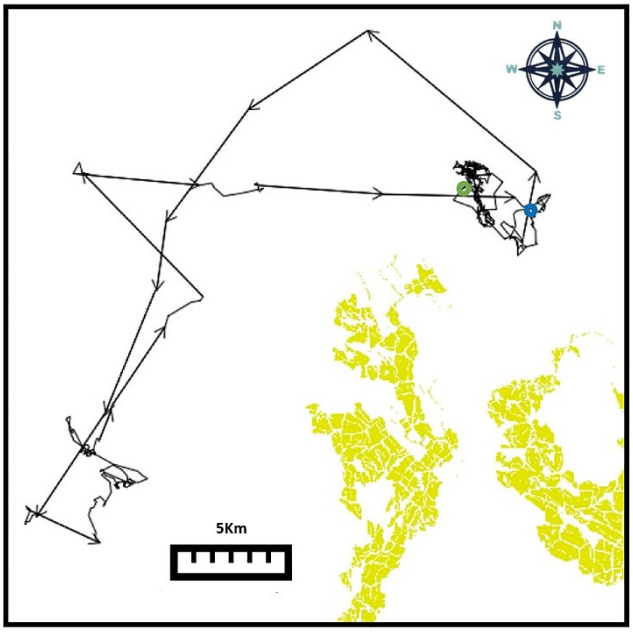


Figure 9: The trajectories of the caribou in the observed time period where the blue circle represents the starting point and the green circle represents the end of the observed trajectory. The area highlighted in yellow represents the areas in the environment with regenerating cuts.

The locations which the caribou visits between times 0 and time $t-1$ at time t are grouped into clusters. For example, cluster 1 refers to the set of locations which were visited by the animal between times 0 and t_1. Within these clusters, the centroids are calculated and the cluster whose centroid is closest to the animal's current position is used in order to compute the direction x_{center}.

A two-state model with $K = 2$ which was proposed and is fitted to the caribou data. For this model, state 1 represents the "travelling" state where larger distances are travelled whereas state 2 represents the "encamped" state where minimal movement is noticed. The explanatory variables considered in this study are directional persistence, x_{cut} and x_{center}. These explanatory variables are represented by the parameters $\kappa^{(k)}_{\text{persist}}$, $\kappa^{(k)}_{\text{cut}}$ and $\kappa^{(k)}_{\text{center}}$ in the model respectively where $k = 1, 2$ represents the current state of the model. In the model, n_k and q_k denote the size and probability of the negative binomial distribution of the waiting time of state k. Both q_1 and q_2 form the transition probability matrix for the hidden Markov chain denoted as:

$$P = \begin{bmatrix} 1-q_1 & q_1 \\ q_2 & 1-q_2 \end{bmatrix}$$

The criterion used for comparison between the various models are AIC, BIC and likelihood. The parameter estimates with their corresponding standard errors (S.E) are shown for each of the states in the model. The first directional model to be presented is the von Mises distribution. The results will be shown together with the four

distance models. The first two to be presented in Table 3 is the case where the distance is fitted using the Gamma distribution and Weibull distribution. In Table 3, $\phi_1^{(k)}$ and $\phi_2^{(k)}$ represent the shape and scale parameters of the Gamma distribution and the Weibull distribution for state k respectively. These two parametric considerations reflects that of Nicosia et al. [32] and is used as a basis of comparison for the distance models considerations. In Tables 3 and 4, the "*" indicates that these parameters are insignificant at a 10% level of significance.

		von Mises (Markov)			
		Gamma		Weibull	
		Estimate	S.E	Estimate	S.E
State 1	q_1	0.2788	0.0891	0.2716	0.0859
	n_1	1	-	1	-
	$\kappa_{\text{persist}}^{(1)}$	1.2666	0.3328	1.2225	0.3326
	$\kappa_{\text{cut}}^{(1)}$	0.1601*	0.3013	0.1582*	0.2929
	$\kappa_{\text{center}}^{(1)}$	0.3733*	0.2994	0.3298*	0.2895
	$\phi_1^{(1)}$	0.6478	0.1373	0.7179	0.0954
	$\phi_2^{(1)}$	3.0445	0.8451	1.425	0.5103
State 2	q_2	0.0231	0.0231	0.0258	0.0258
	n_2	1	-	1	-
	$\kappa_{\text{persist}}^{(2)}$	0.0274*	0.0618	0.0183*	0.0644
	$\kappa_{\text{cut}}^{(2)}$	0.1454	0.0698	0.146	0.0712
	$\kappa_{\text{center}}^{(2)}$	0.259	0.0694	0.2612	0.0705
	$\phi_1^{(2)}$	1.2626	0.0774	1.1143	0.0478
	$\phi_2^{(2)}$	0.1351	0.012	0.1757	0.0093
	Likelihood	-794.91	-	-798.301	-
	AIC	1613.8199	-	1620.603	-
	BIC	1666.9183	-	1673.701	-
	Run Time	76.01s	-	92.69s	-

Table 3: The estimation of the model with the von Mises distribution for the directional component and the **Gamma** and **Weibull** distributions as the distance component

The consideration of the von Mises distribution for the directional component with the power Lindley and Gumbel distributions as the distance component is now presented in Table 4. Herein, $\phi_1^{(k)}$ and $\phi_2^{(k)}$ represent the scale and shape parameters of the power Lindley distribution and the location and scale parameters for the Gumbel distributions respectively for state k.

		von Mises (Markov)			
		Power Lindley		Gumbel	
		Estimate	S.E	Estimate	S.E
State 1	q_1	0.0734	0.0409	0.9484	0.1879
	n_1	1	-	1	-
	$\kappa^{(1)}_{\text{persist}}$	0.1872	0.0707	0.9161	0.4563
	$\kappa^{(1)}_{\text{cut}}$	-0.0852	0.0883	-0.8856	0.4668
	$\kappa^{(1)}_{\text{center}}$	0.1868	0.0743	0.9493	0.4168
	$\phi^{(1)}_1$	3.3596	0.1758	0.2099	0.0123
	$\phi^{(1)}_2$	0.7012	0.0287	0.1407	-
State 2	q_2	0.2709	0.2709	0.2824	0.2824
	n_2	1	-	1	-
	$\kappa^{(2)}_{\text{persist}}$	-0.7817*	0.4073	-0.0034*	0.1054
	$\kappa^{(2)}_{\text{cut}}$	4.5847	1.5542	0.3707	0.1118
	$\kappa^{(2)}_{\text{center}}$	3.9761*	1.2426	0.1774*	0.1066
	$\phi^{(2)}_1$	4.1603	0.3915	0.2109	0.0066
	$\phi^{(2)}_2$	0.7602	0.0264	0.1388	-
	Likelihood	-1005.93	-	-523.568	-
	AIC	2035.856	-	1071.135	-
	BIC	2088.954	-	1124.234	-
	Run Time	65.65s	-	72.53s	-

Table 4: The estimation of the model with the von Mises distribution for the directional component and the **power Lindley** and **Gumbel** distributions as the distance component

Based on the results from the tables above, the Gumbel distribution seems to perform the best amongst the other distance considerations based on the BIC values. However, as mentioned earlier, this should be erred with caution as the pdf may not always lie on the positive real line due to pdf tails which extend to the left. However, based on the estimates above in Table 4, both parameter values are positive indicating that the pdf is positive in this particular modelling scenario. This may not necessarily hold true across different datasets as the modelling of animal movement is a data-driven process. This illustrates that the Gumbel distribution could be a useful tool and valid for consideration in the distance modelling of animal movement.

The Gamma distribution performed the second best for modelling the distances, followed by the Weibull and power Lindley distributions in terms of BIC values. It is interesting that although the power Lindley distribution is a mixture of the Gamma and Weibull distributions, it seems to have the worst performance. The results of the power Lindley distribution for distance modelling of animal movement together with the von Mises for directional movement will be studied more closely in the next section using a bootstrap procedure.

The second case which is considered is the ssvM distribution for the directional component of the model. The results will be shown together with the four distance models. The notation follows in the same way as for the von Mises results above. The skewness parameter λ will initially need to be set before the respective models are run. To this end, the model was run multiple times using various values ranging from $\lambda = 0.1, \ldots, 0.9$ and used

the corresponding AIC and BIC values to determine which value of λ was most suitable. The circular plot below illustrates that there is unfortunately no skewness present in the directional data, so an arbitrarily small value of $\lambda = 0.1$ is selected, which noted a minor increase in the AIC and BIC values.

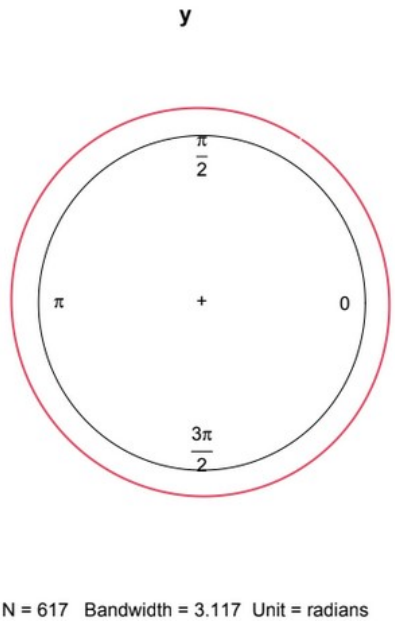


Figure 10: The kernel density plot of y_t

The first to be presented in Table 5 is the case where the distance is fitted using the Gamma and Weibull distributions with the ssvM fitting the directional component with the aforementioned skewness parameter set at $\lambda = 0.1$. In Table 5, $\phi_1^{(k)}$ and $\phi_2^{(k)}$ represent the shape and scale parameters of the Gamma and Weibull distributions respectively for state k.

		ssvM (Markov)			
		Gamma		Weibull	
		Estimate	S.E	Estimate	S.E
State 1	q_1	0.2788	0.0891	0.2716	0.0859
	n_1	1	-	1	-
	$\kappa^{(1)}_{\text{persist}}$	1.2666	0.3328	1.2225	0.3326
	$\kappa^{(1)}_{\text{cut}}$	0.1601*	0.3013	0.1582*	0.2929
	$\kappa^{(1)}_{\text{center}}$	0.3733*	0.2994	0.3298*	0.2895
	$\phi^{(1)}_1$	0.6478	0.1373	0.7179	0.0954
	$\phi^{(1)}_2$	3.0445	0.8451	1.425	0.5103
State 2	q_2	0.0231	0.0231	0.0258	0.0258
	n_2	1	-	1	-
	$\kappa^{(2)}_{\text{persist}}$	0.0274*	0.0618	0.0183*	0.0644
	$\kappa^{(2)}_{\text{cut}}$	0.1454	0.0698	0.146	0.0712
	$\kappa^{(2)}_{\text{center}}$	0.259	0.0694	0.2612	0.0705
	$\phi^{(2)}_1$	1.2626	0.0774	1.1143	0.0478
	$\phi^{(2)}_2$	0.1351	0.012	0.1757	0.0093
	Likelihood	-795.4334	-	-798.825	-
	AIC	1614.8669	-	1621.65	-
	BIC	1667.9653	-	1674.748	-
	Run Time	78.39s	-	96.23s	-

Table 5: The estimation of the model with the ssvM distribution for the directional component and the **Gamma** and **Weibull** distributions for the distance component

The consideration of the von Mises distribution for the directional component with the power Lindley and Gumbel distributions as the distance component is now presented in Table 6. Herein, $\phi^{(k)}_1$ and $\phi^{(k)}_2$ represent the scale and shape parameters of the power Lindley distribution and the location and scale parameters for the Gumbel distributions respectively for state k.

		ssvM (Markov)			
		Power Lindley		Gumbel	
		Estimate	S.E	Estimate	S.E
State 1	q_1	0.2942	0.1128	0.8344	0.2177
	n_1	1	-	1	-
	$\kappa^{(1)}_{\text{persist}}$	3.4148	1.5498	0.5617	0.3463
	$\kappa^{(1)}_{\text{cut}}$	-4.0876	1.8133	-0.5757	0.3541
	$\kappa^{(1)}_{\text{center}}$	1.014	0.8774	0.7412	0.3215
	$\phi^{(1)}_1$	3.1673	0.3069	0.2134	0.0106
	$\phi^{(1)}_2$	0.7712	0.0907	0.1416	-
State 2	q_2	0.0114	0.0114	0.3723	0.3723
	n_2	1	-	1	-
	$\kappa^{(2)}_{\text{persist}}$	0.0591*	0.0585	0.0021*	0.1446
	$\kappa^{(2)}_{\text{cut}}$	0.2478	0.0654	0.4198	0.1481
	$\kappa^{(2)}_{\text{center}}$	0.3326*	0.0682	0.1467*	0.1485
	$\phi^{(2)}_1$	3.5031	0.1794	0.2095	0.0069
	$\phi^{(2)}_2$	0.7075	0.0288	0.1382	-
	Likelihood	-1012.17	-	-524.586	-
	AIC	2048.35	-	1073.171	-
	BIC	2101.448	-	1126.27	-
	Run Time	67.19s	-	74.96s	-

Table 6: The estimation of the model with the ssvM distribution for the directional component and the **power Lindley** and **Gumbel** distributions for the distance component

Based on the results presented in Tables 5 and 6, similar results have been observed as the von Mises directional component in Tables 3 and 4. The BIC values differ slightly across all distributions with the Gumbel distribution still providing the best result, with the other distance distributions performing in the same order as earlier. This illustrates the versatility of the ssvM in this modelling context, noting only minor increases in computational times across all models. The same significances across parameters are noted for both the von Mises and ssvM modelling scenarios. It should be noted that this ssvM model will certainly perform better than the von Mises distribution in the presence of skewed data, but detailed investigations in this matter is beyond the scope of this study.

For the first state of the model, large estimates were obtained across all models for the average distance travelled by the animal i.e. $\hat{\phi}^{(k)}_1\hat{\phi}^{(k)}_2$ except the Gumbel distribution which seems to be an outlier in its performance. An interesting point to highlight is that the power Lindley obtained similar distances to the Gamma distribution in state one, which reinforces the intrigue to further investigate it's performance in the next section. The caribou exhibit positive directional persistence in their movement illustrated by $\kappa^{(1)}_{\text{persist}}$ across the models. The two environmental factors considered in the model had varying results across the models. For both the Weibull and Gamma models, $\kappa^{(1)}_{\text{cut}}$ and $\kappa^{(1)}_{\text{center}}$ had positive estimates with fairly large standard errors. In contrast, the power Lindley and Gumbel models noted a negative estimate for $\kappa^{(1)}_{\text{cut}}$ and a positive estimate for $\kappa^{(1)}_{\text{center}}$ with fairly large standard errors. The Gumbel and power Lindley seem to be a more logical choice in this case, since Figure 9

illustrates that the animal never reaches the regenerating cuts area. These large standard errors could be due to the fact that there is limited amounts of data which is available for state 1. In order to obtain the average speed of at which the animal moves in this state, $\hat{\phi}_1^{(1)}\hat{\phi}_2^{(1)}/4$ is computed. The Gamma distribution reported a distance of approximately 494m per hour, with the power Lindley reporting 611m per hour, the Weibull reporting 255m per hour and lastly, the Gumbel reporting 7m per hour. The Gumbel distribution clearly doesn't provide a reasonable estimation in this scenario as previously highlighted by the average distances obtained.

For the second state of the model, the animal is in its "encamped" state in which it is primarily stationary. Smaller estimates were found for the average speeds with the Gamma reporting 44m per hour and the Weibull reporting 49m per hour. The power Lindley and Gumbel distributions were quite different with their results, noting 357m per hour and 7.4m per hour respectively. The directional persistence $\kappa_{\text{persist}}^{(2)}$ was found to be insignificant in this state across all models which is a sensible result given the stationary movement state of the caribou. Both of the environmental factors considered in the model were found to be significant. It is interesting to note that $\kappa_{\text{cut}}^{(2)}$ was significant and positive for this state across all models when the caribou never reaches this area as illustrated by Figure 9. This may be a fortuitous relationship given the north-south movement trajectory of the caribou and the southern location of the regenerating tree cuts.

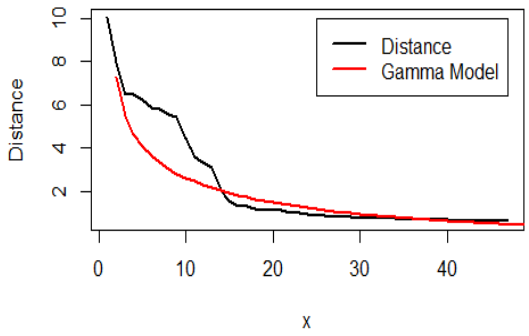
The stationary distribution of the latent fitted Markov chain is in the first state with the corresponding probability $\frac{\hat{q_2}}{\hat{q_2}+\hat{q_1}}$. This probability can also show the number of sightings the caribou was observed travelling. Using the von Mises directional results, the stationary distributions are summarised in the table below:

	Probability	No. of travelling States
Gamma	0.0765	47
Power Lindley	0.1126	69
Weibull	0.0868	54
Gumbel	0.2294	142

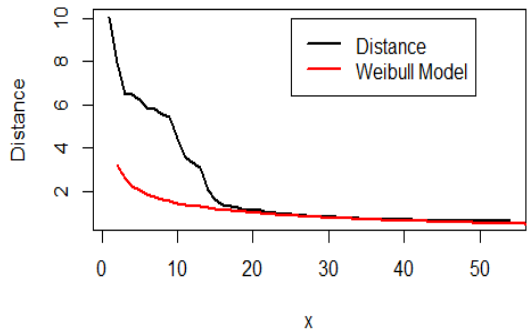
Table 7: Stationary distribution of the latent fitted Markov chain for state 1

The number of travelling states observed for the caribou expressed in the table above provides an explanation to the low precision and high standard errors found across all models in the first state. The Gumbel distribution found the largest amount of travelling states yet, provided the smallest estimate for distance travelled. In contrast, the power Lindley distribution found a larger amount of travelling states than the Gamma and Weibull counterparts and provided a larger estimate for distance travelled, which is a logical result.

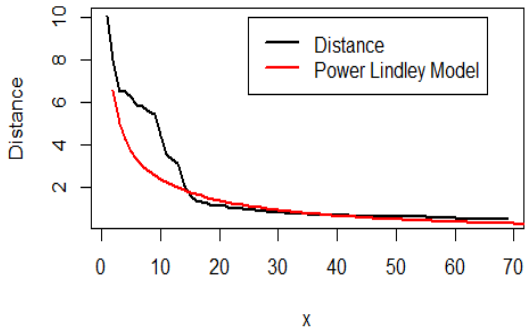
Using the stationary distributions in Table 7 and parameter estimates obtained in Tables 3 and 4, the following plots in Figure 11 were made to illustrate how the various distributions fitted the distances for state 1:



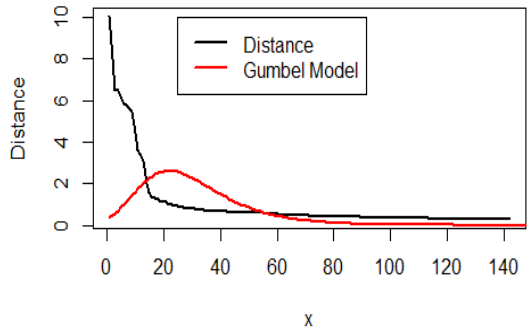
(a) Gamma distance fit



(b) Weibull distance fit



(c) Power Lindley distance fit



(d) Gumbel distance fit

Figure 11: The various distance component fits for the model for state 1

Figure 11 illustrates that the power Lindley and Gamma distributions fit the distances relatively well with the Weibull distribution providing a slightly worse fit. The worst is clearly the Gumbel distribution, where the shape of the pdf hardly resembles that of distances being modelled. It should once again be noted that the modelling of animal movement is a data driven process. Although the Gumbel distribution has the worst distance modelling performance, it might be better suited to modelling shorter distances.

The state of the caribou's position at time t may be identified using the smoothed probabilities $P(S_{tk} = 1|\mathscr{F}_T; \hat{\theta}_{\text{MLE}}), t = 1, \ldots, T,\ k = 1, 2$ which is computed in the smoothing step of the filtering-smoothing algorithm within the E-step of the EM algorithm. This is visually depicted for the Weibull distribution with a von Mises directional component in the following figure:

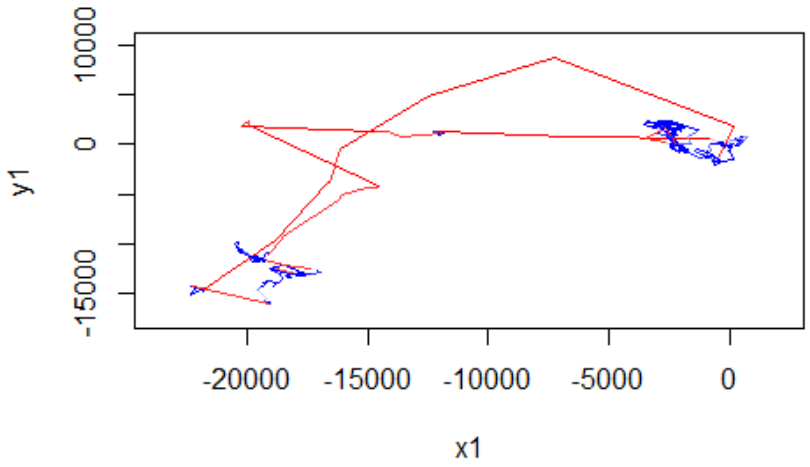


Figure 12: The hidden state probabilities illustrated for each time step of the caribou's movement trajectory.

In Figure 12, the red colour represents being in state 1 i.e. $P(S_{t1} = 1|\mathscr{F}_T; \hat{\theta}_{\text{MLE}}) = 1$ and the blue colour represents being in state 2 i.e. $P(S_{t2} = 1|\mathscr{F}_T; \hat{\theta}_{\text{MLE}}) = 1$. The caribou movement was clearly influenced by various factors which include directional persistence, which only affected their movement in state 1. Furthermore, the directional analysis revealed that the two environmental factors considered, x_{cut} and x_{center}, influenced their movement which enabled the generation of plots such as Figure 12 which may be used to identify areas of movement for the different movement states of the caribou [32]. This model and its various alternate parametric considerations provide a more generalised set of modelling tools and results for animal movement modelling.

3.4 Bootstrap

Bootstrapping is a popular resampling method which is typically used to estimate statistics on a population by sampling from the population dataset with replacement. This is the simplest method to obtain the sampling distribution of sample statistics [18].

The procedure of bootstrapping stems from the idea of resampling from the original population many times with replacement, where each resample is the same size as the original random sample [18]. The detail of sampling with replacement is of utmost importance in bootstrap sampling, because it permits the chance of drawing any observation more than once. If this were not the case, the bootstrap sample would ultimately contain the population again in each sample with its observations in a different order. The desired statistic is calculated for each resample and the distribution of all the collective resample statistics is called the bootstrap distribution [18]. This bootstrap distribution provides many insights into the shape, spread and behaviour of the sampling distribution of the statistic

being calculated.

In the previous section, the power Lindley distribution was considered as an alternative for the distances in animal movement modelling. Upon closer inspection of the obtained estimates, the power Lindley distribution proved to be a unique consideration in the context of animal movement modelling in light of its parametric performance and interpretability to the Gamma and Weibull distributions. The motivation for further investigation into this model is the construction of the power Lindley distribution, which consists of a mixture between the Gamma and Weibull distributions as well as the appeal of increased computational efficiency as illustrated by the lower model run times in Table 4. In this section, a bootstrap procedure is performed using the von Mises distribution for the directional component to obtain the sampling distribution for the parameters of the power Lindley distribution from the fitted model. It should be noted that both the directional and distance components were jointly modelled, but only the distance component fits are illustrated. This bootstrap study aims to "mimic" real data for illustrative purposes which will be able to highlight the potential capabilities of the power Lindley distribution.

The algorithm for the bootstrap of the power Lindley distribution is outlined below:

1. Generate $m = 50, 100, 200, 300$ samples of size $n = 617$ i.e. the size of the dataset with replacement from the original dataset.

2. Calculate and store the estimated parameters of the power Lindley obtained from fitting the model i.e $\phi_1^{(k)}$ and $\phi_2^{(k)}$ defined as in the previous section for states $k = 1, 2$ as well as the AIC, BIC and log-likelihood values for each resample.

3. Calculate the bootstrap distribution and sample summary statistics for each of the parameters.

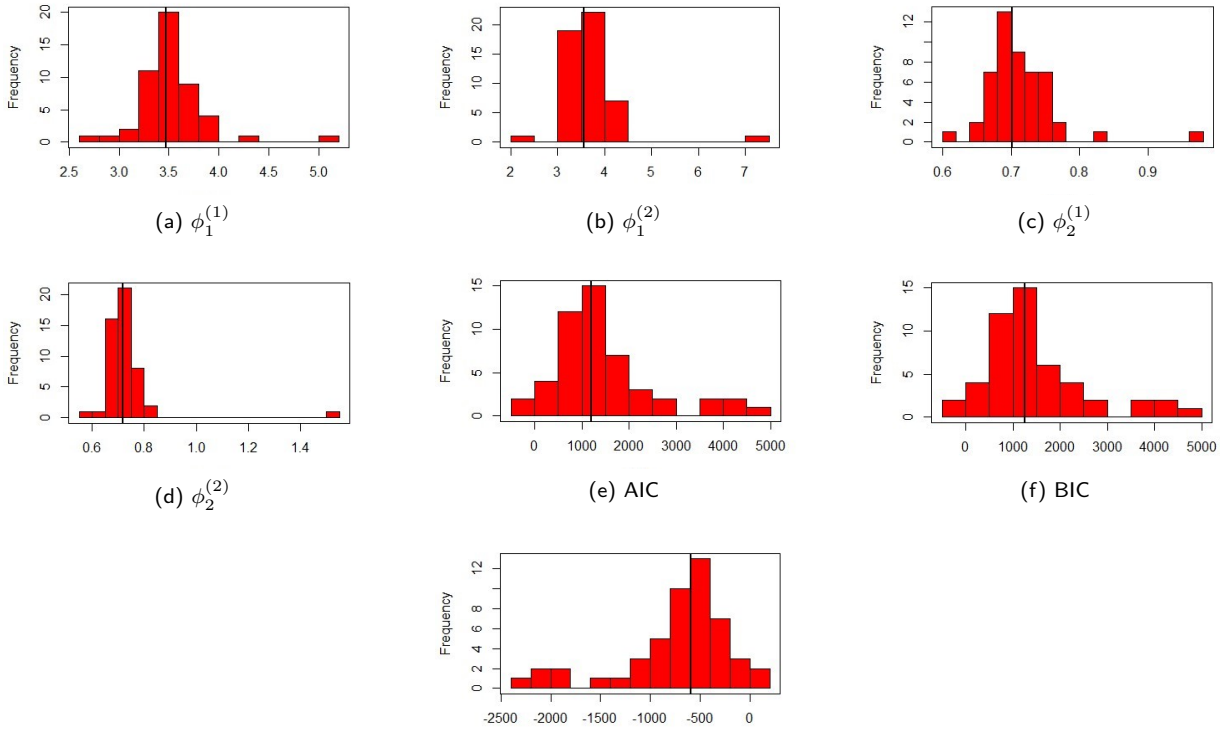
4. Plot a histogram for the bootstrap distribution with a vertical reference line for the median.

Across all the histograms, the solid black vertical reference line represent the median value of the sample statistic. The median is considered to be a more robust measure of location than the mean due to the skewed nature of some of the bootstrap distributions.

The results for the summary statistics and bootstrap distributions are shown in Table 8 for $m = 50$:

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
$\phi_1^{(1)}$	2.64	3.32	3.47	3.52	3.65	5.08
$\phi_1^{(2)}$	2.01	3.43	3.56	3.64	3.69	7.16
$\phi_2^{(1)}$	0.61	0.69	0.70	0.71	0.73	0.97
$\phi_2^{(2)}$	0.58	0.69	0.72	0.73	0.74	1.53
AIC	-166.79	840.84	1208.68	1453.45	1797.92	4518.81
BIC	-113.69	893.94	1261.78	1506.55	1851.01	4571.90
LL	-2247.40	-886.96	-592.34	-714.72	-408.42	95.39

Table 8: Summary statistics of the bootstrap distributions obtained for $m = 50$



(a) $\phi_1^{(1)}$ (b) $\phi_1^{(2)}$ (c) $\phi_2^{(1)}$

(d) $\phi_2^{(2)}$ (e) AIC (f) BIC

(g) LL

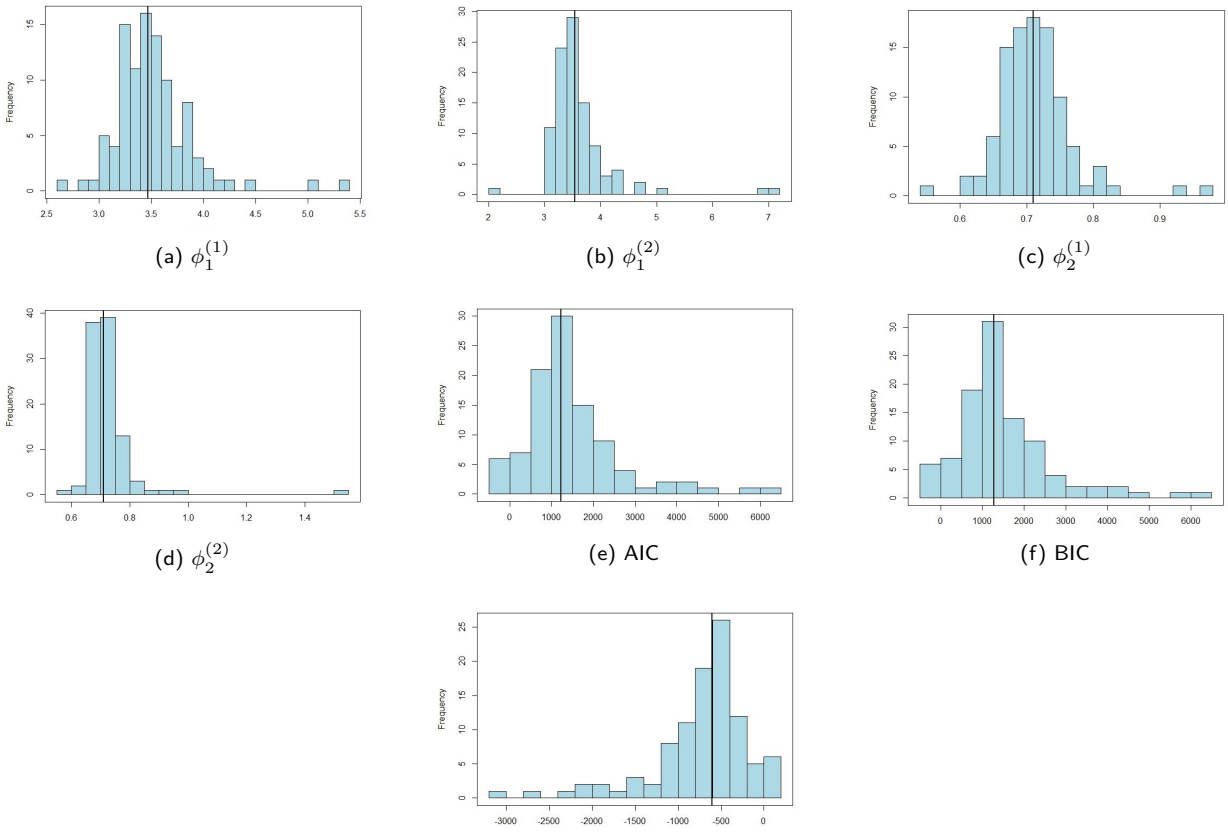
Figure 13: Bootstrap histograms for $m = 50$

When $m = 50$, the number of resamples is still relatively small. These values resemble the results found in the previous section, however, a large spread of values is noted for the estimates. It is interesting to note that the AIC and BIC values have decreased in comparison to the results shown in Table 4. Furthermore, the bootstrap distributions of the metrics are all right-skewed, with the exception of the likelihood bootstrap distribution. However, this might be due to the relatively small sample size taken and the large amount of variability exhibited by the bootstrap results. This can be further investigated when the sample size increases.

The results for the summary statistics and bootstrap distributions are shown in Table 9 for $m = 100$:

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
$\phi_1^{(1)}$	2.64	3.29	3.47	3.53	3.68	5.38
$\phi_1^{(2)}$	2.01	3.32	3.54	3.62	3.70	7.16
$\phi_2^{(1)}$	0.54	0.68	0.71	0.71	0.74	0.97
$\phi_2^{(2)}$	0.58	0.68	0.71	0.73	0.74	1.53
AIC	-363.29	848.22	1231.16	1461.97	1833.92	6078.44
BIC	-310.20	901.32	1284.26	1515.07	1887.02	6131.54
LL	-3027.22	-904.96	-603.58	-718.98	-412.11	193.65

Table 9: Summary statistics of the bootstrap distributions obtained for $m = 100$



(a) $\phi_1^{(1)}$ (b) $\phi_1^{(2)}$ (c) $\phi_2^{(1)}$

(d) $\phi_2^{(2)}$ (e) AIC (f) BIC

(g) LL

Figure 14: Bootstrap histograms for $m = 100$

These values obtained when $m = 100$ resemble the results found in the previous section, however, a large spread of values is once again noted for the estimates. It is interesting to note that the AIC and BIC values have decreased in comparison to the results shown in Table 4 but increased slightly from when $m = 50$. Furthermore, the bootstrap distributions all have the same skewness properties exhibited in the previous sample, where the histograms are right-skewed with the exception of the likelihood bootstrap distribution. Large amounts of variability exhibited are exhibited by the results once again which is shown by the spread of values obtained in Table 9. An interesting

observation is the range of values from the minimums of the AIC and BIC to the third quartile. The minimum values obtained are clearly extremely low, coupled with meaningful parameter estimates which are interpretable. These AIC and BIC values range from approximately -300 to the 1880, which is still comparable to the results obtained by the other distributions in Tables 3 and 4.

The results for the summary statistics and bootstrap distributions are shown in Table 10 for $m = 200$:

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
$\phi_1^{(1)}$	1.97	3.31	3.51	3.60	3.83	5.38
$\phi_1^{(2)}$	2.01	3.32	3.52	3.59	3.73	7.16
$\phi_2^{(1)}$	0.54	0.68	0.71	0.72	0.75	1.00
$\phi_2^{(2)}$	0.56	0.68	0.71	0.72	0.74	1.53
AIC	-633.49	945.82	1345.71	1605.15	2031.15	7865.78
BIC	-580.39	998.92	1398.81	1658.25	2084.25	7918.88
LL	-3920.89	-1003.58	-660.86	-790.57	-460.91	328.74

Table 10: Summary statistics of the bootstrap distributions obtained for $m = 200$

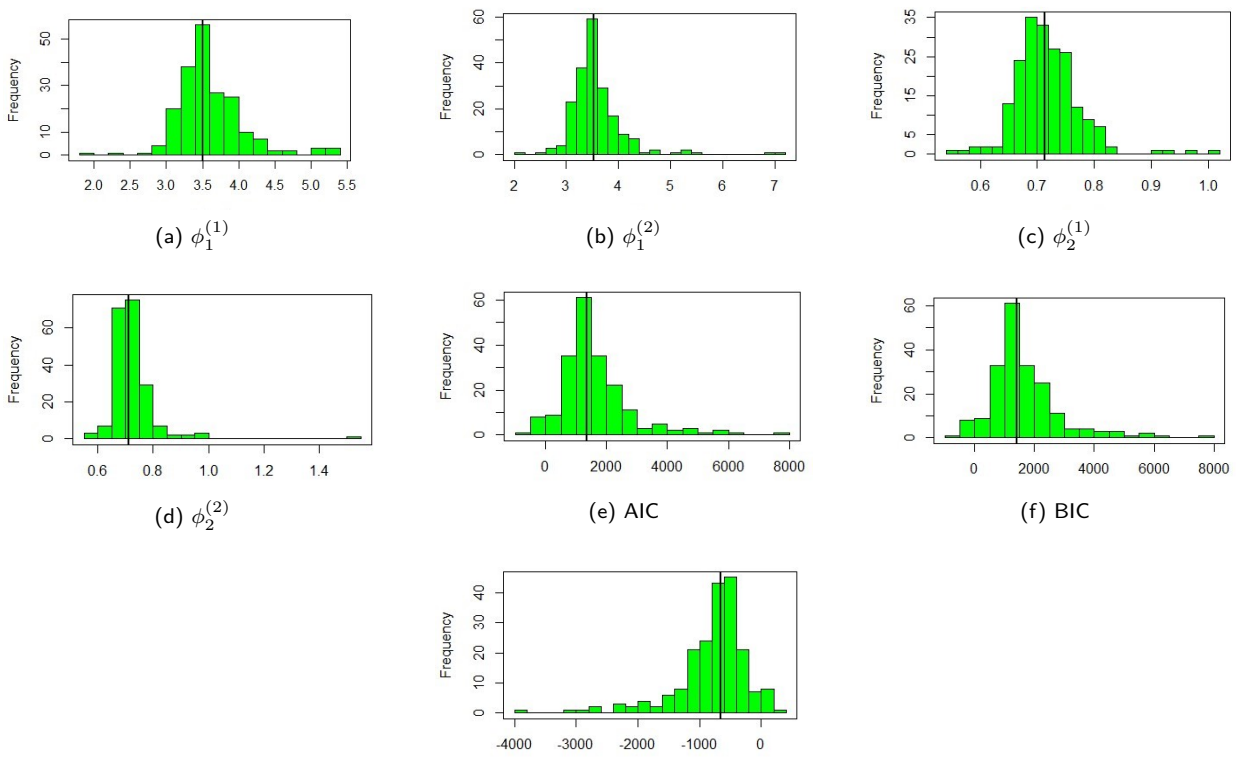


Figure 15: Bootstrap histograms for $m = 200$

The values obtained for $m = 200$ follow the same trend of obtaining similar values as the original model, which is to be expected. Another increase in the AIC and BIC value is noted but it is still significantly lower than

the values obtained in Table 4. The same skewness properties are exhibited as the previous two samples, which illustrates consistency in the bootstrap results. The power Lindley parameter estimates remain fairly consistent across the sample sizes to the original values obtained.

The results for the summary statistics and bootstrap distributions are shown in Table 11 for $m = 300$:

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
$\phi_1^{(1)}$	1.97	3.32	3.50	3.60	3.80	6.60
$\phi_1^{(2)}$	1.99	3.29	3.51	3.57	3.73	7.16
$\phi_2^{(1)}$	0.54	0.69	0.71	0.72	0.75	1.07
$\phi_2^{(2)}$	0.52	0.68	0.71	0.72	0.75	1.53
AIC	-633.49	937.37	1371.99	1652.03	2090.86	7865.78
BIC	-580.39	990.47	1425.09	1705.13	2143.96	7918.88
LL	-3920.89	-1033.43	-674.00	-814.02	-456.68	328.74

Table 11: Summary statistics of the bootstrap distributions obtained for $m = 300$

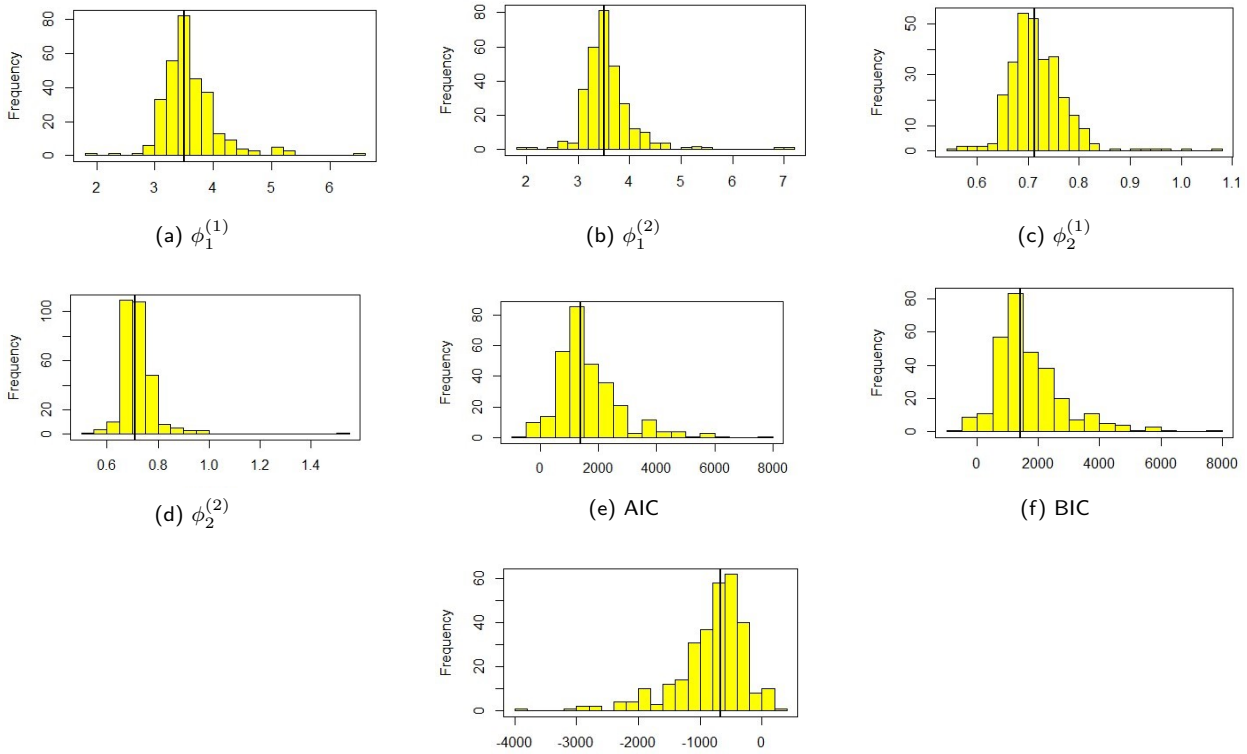


Figure 16: Bootstrap histograms for $m = 300$

The parameter values remain fairly similar across all bootstrap samples for the various sample sizes. These parameter values of $\phi_1^{(k)}$ and $\phi_2^{(k)}$ for $k = 1, 2$ are similar to the values obtained by Table 4. Drawing comparisons between these bootstrap results against the original results reveals that the AIC and BIC values are higher than

the Gumbel distribution, but lower than the other models. The minimum values obtained by the bootstrap further emphasise the potential of the power Lindley distribution as a preferable model for distance modelling for a variety of samples and across different sample sizes. The minimum values and first quartile values obtained for the estimates as well as the AIC and BIC measures indicate that in certain circumstances, the power Lindley can strongly outperform the parametric counterparts considered in this study whilst retaining interpretability and computational efficiency. This is an important result to note, given the appealing features of the power Lindley's modelling performance and results previously discussed. The results obtained by this bootstrap procedure reinforce that the power Lindley distribution is an ideal candidate for the modelling of distances within animal movement modelling for this specific data modelling scenario. The performance of the power Lindley for modelling distances in animal movement illustrates better performance than the Weibull and Gamma distributions, whilst still remaining true to the movement nature of the caribou. It is still important to note that the modelling of animal movement is a data driven process and no distribution or model will be a universally accepted candidate for all modelling instances.

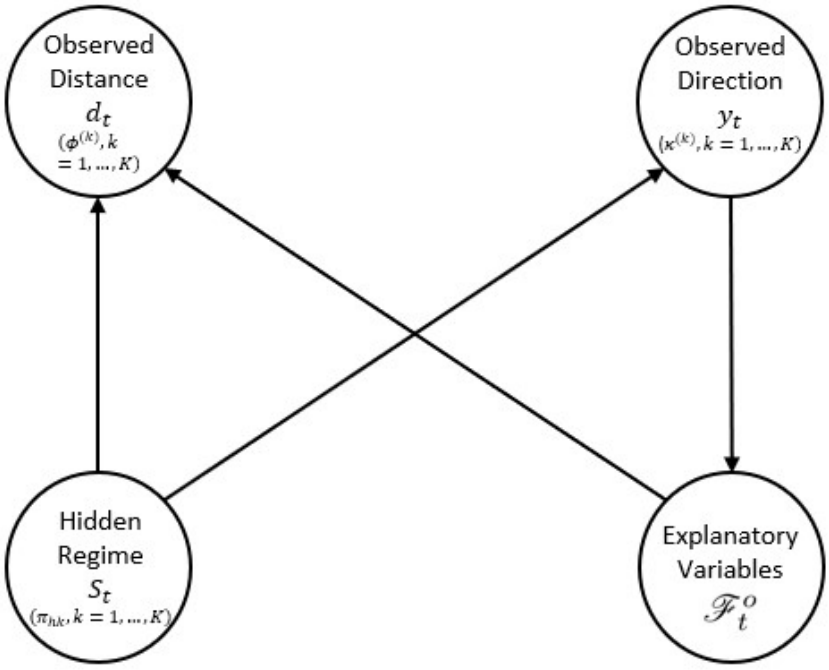


Figure 17: The dependence structure of the model within the BCRW

Figure 17 provides a visual summary of the model and its respective components. This summarises the dependence structure within the model between the explanatory variables, observed bivariate process (39) and the hidden process.

4 Conclusion

The modelling of animal movement is a data dependent process. A variety of alternative parametric models were considered for the modelling of animal movement in this study. The von Mises and ssvM distributions were considered for the directional component of the multi-state BCRW model with the Gamma, Weibull, power Lindley and Gumbel distributions being considered for the distance component. This model comprises of a two-state BCRW model with hidden Markovian states to account for the movement behaviour of the animal. This model has the ability to account for multiple movement behaviours as well as the animal's response to environmental factors. In this modelling scenario, two environmental factors were accounted for together with one directional explanatory variable. This model was estimated using an EM algorithm with a filtering-smoothing algorithm embedded within the E-step of the EM algorithm.

The modelling results illustrated that the von Mises distribution for the directional component together with the Gumbel distribution for the distance component of the model provided the best results using BIC as the measure of performance. However, upon closer inspection of the parameter estimates obtained, the Gumbel distribution failed to adequately provide meaningful parametric results in terms of animal movement modelling for this specific dataset. The Gamma distribution performed well in comparison to the Weibull and power Lindley counterparts which was detailed by Nicosia et al. [32], however, interpretation of the power Lindley estimates revealed similar parametric results to that of the Gamma and Weibull. Despite having a higher BIC value, the power Lindley's nature of being a mixture of Gamma and Weibull distributions warranted further investigation into its performance via a bootstrap simulation study. This bootstrap simulation study revealed that as the number of samples increased, the median BIC value decreased and was ultimately lower than the Weibull and Gamma distribution's BIC values. This would therefore indicate that the power Lindley is a valid consideration for modelling an animal's distance in animal movement models in terms of tractability, computational efficiency and model performance.

Animal poaching is a current international issue. Animal movement models which are able to account for multiple types of animal behaviour, track movement patterns and are easily interpretable will be a valuable asset to ecologists and conservationists. This modelling framework with the alternative parametric direction and distance considerations might provide a foundation for future animal movement models to improve on in both accuracy and efficiency.

There are multiple possible options for further extensions of the methods presented in this study. The consideration of the generalised von Mises distribution and Exponentiated Cardioid for directional movement modelling of animal movement. A variety of other distributions or mixture models could be considered for the distance modelling of the animals to extend the current range of considered distance models. Random effects could be introduced into the model to account for the simultaneous analysis of multiple individuals but will require a new

numerical procedure to estimate the model, which could prove to be difficult. Various special cases can be considered to extend the consensus model construction, such as increasing the number of hidden states, changing model assumptions and implementing multi-state modelling frameworks. A semi-Markov approximation for the hidden Markov model can be implemented into this modelling framework. A contamination study can be performed where the data is intentionally polluted with random errors to test the robustness and flexibility of the model when faced with data of this nature.

5 Appendix A

5.1 Linear - Circular Regression

Circular data has a periodic nature which fundamentally differs from linear data. Measurements on a circle at $360°$ and $0°$ represent the same direction whereas they would be on opposite ends of a linear scale [10]. Early findings on linear-circular regression were based on the assumption that the circular response θ follows the von Mises distribution (1) given predictors $X = \mathbf{x}$. Considering the von Mises distribution, the mean μ is regressed over observations $\mathbf{x_i}$ using a circular link function $h(\cdot)$ [39],

$$\mu_i = \mu_0 + h(\mathbf{x_i}). \tag{59}$$

The circular link function has had many proposed functions, such as $h(\mathbf{x}) = \beta'\mathbf{x}$ [16] and $h(\mathbf{x}) = 2\pi F(\mathbf{x})$ [20] where $F(\mathbf{x})$ denotes the marginal function of $\mathbf{x}$. Maximum likelihood approaches were proposed to estimate the regression parameters, but it proved to be difficult to optimise the likelihood functions due to multimodality and unidentifiability of parameters [39].

An alternative approach would be to use non-parametric smoothing which involves finding an unknown regression function $\mu(\cdot)$ minimising the angular risk function denoted by [39],

$$E[1 - \cos(\theta - \mu(X))|X = \mathbf{x}]. \tag{60}$$

The minimiser of risk is $\tan^{-1}(s(\mathbf{x}), c(\mathbf{x}))$, where $s(\mathbf{x}) = E[\sin(\theta)|X = \mathbf{x}]$ and $c(\mathbf{x}) = E[\cos(\theta)|X = \mathbf{x}]$. Non-parametric estimates of $s(\cdot)$ and $c(\cdot)$ are achieved using locally weighted regression over $\sin(\theta)$ and $\cos(\theta)$ respectively across the individual observations, where the estimates are then substituted into the minimiser of the risk therefore yielding the estimate of $\mu(\mathbf{x})$ [39].

The treatment of the circular responses as the projection of unobserved bivariate normal variables on the unit circle is an alternative approach to regression. This method is defined as [36],

$$\theta = \tan^{-1}\left(\frac{y_{i2}}{y_{i1}}\right) \text{ for } y_{i1} > 0 \tag{61}$$

where (y_{i1}, y_{i2}) represents a bivariate normal vector with a covariance of $\mathbf{\Sigma}$ and respective mean $\mu_\mathbf{i} = \boldsymbol{\nu}'\boldsymbol{x_i}$. For this approach, $\mathbf{\Sigma} = \boldsymbol{I}$ where $\boldsymbol{I}$ denotes the identity matrix is fixed where and the EM algorithm is used to estimate $\boldsymbol{\nu}$ [36][39].

The approach of a wrapped distribution could also be used, whereby a circular response θ is regarded as the result of the modulo operation on a real random variable. When the real variable follows a pdf denoted by $f(\cdot)$,

the corresponding circular response follows a wrapped distribution with the pdf [39]

$$f_w(\theta) = \sum_{Z=-\infty}^{\infty} f(\theta + 2Z\pi + \pi). \tag{62}$$

This pdf is achieved by wrapping the aforementioned pdf $f(\cdot)$ around the circumference of a unit circle. This pdf is popularly used to describe a probability distribution for a circular random variable.

A linear model for the von Mises distributions in the two-dimensional case is now introduced. Suppose a sample of M observations $(\mathbf{x_1}, \mathbf{t_1}), \ldots, (\mathbf{x_M}, \mathbf{t_M})$ is obtained from (2) with $\kappa > 0$, the $i-th$ mean vector having the form [16]

$$\boldsymbol{\nu'} = [\cos(\alpha_0 + \alpha' \mathbf{t_i}), \sin(\alpha_0 + \alpha' \mathbf{t_i})], \tag{63}$$

where $\alpha' \mathbf{t_i} = \alpha_1 t_{1i} + \cdots + \alpha_q t_{qi}$ and the total number of distinct t_i's exceed q where q denotes the number of parameters being fitted to the model. In the case where the measurement errors of the $\mathbf{x}$'s are negligibly small, the log-likelihood is

$$L(\alpha_0, \ldots, \alpha_q) = \text{constant} + \kappa \sum_{i=1}^{M} \mathbf{x_i'} \mathbf{v_i} \tag{64}$$

The maximum likelihood solutions for the parameters $\hat{\alpha_0}, \hat{\alpha_1}, \ldots, \hat{\alpha_q}$ are obtained as the solutions to the system which consists of $q+1$ equations

$$\frac{\partial L(\hat{\alpha_0}, \ldots, \hat{\alpha_q})}{\partial \hat{\alpha_i}} = 0, \quad i = 0, 1, \ldots, q. \tag{65}$$

The result that (65) equals zero implies that [16]

$$\tan(\hat{\alpha_0}) = \left[\sum_{i=1}^{M} \mathbf{x_i'} \begin{pmatrix} -\sin(\hat{\alpha'} \mathbf{t_i}) \\ \cos(\hat{\alpha'} \mathbf{t_i}) \end{pmatrix} \right] \left[\sum_{i=1}^{M} \mathbf{x_i'} \begin{pmatrix} \cos(\hat{\alpha'} \mathbf{t_i}) \\ \sin(\hat{\alpha'} \mathbf{t_i}) \end{pmatrix} \right]^{-1}, \tag{66}$$

where $\hat{\alpha_0}$ may be obtained as a function of $\hat{\alpha_1}, \ldots, \hat{\alpha_q}$. An explicit solution to (64) is not possible, so an iterative procedure such as the Gauss-Newton algorithm may be used. To make use of the Gauss-Newton method [16], the Taylor expansion would be required.

A natural model for regression of a linear variable X on an angular variable θ is [28]:

$$X|\theta \sim N(\alpha + \beta_1 \cos(\theta) + \beta_2 \sin(\theta), \sigma^2) \tag{67}$$

The above is multiple linear regression of X on (cos(θ),sin(θ)) where θ is the angular variable.

Suppose that Θ is a circular response variable measured at values $x_1, ..., x_n$ of a k-dimensional explanatory variable X. When dealing with a one-dimensional explanatory variable, the proposition of wrapping the line onto

the circle leads to Gould's regression model [16] where

$$\Theta_i \sim M(\mu + \beta x_i, \kappa), \qquad i = 1, ..., n. \tag{68}$$

Note that μ, β and κ are unknown parameters and Θ_1,..., Θ_n are independent. This model is the circular response regression model. It has the disadvantage of having a likelihood that has infinitely many maxima of comparable size [28] and a non-identifiable β when the x_i's are equally spaced. This creates the problem of finding the maximum likelihood estimators. This problem is caused by the helical regression function $h(x) = \beta x (\text{mod} 2\pi)$ which wraps the real line around the circle infinitely many times. To avoid this problem, link functions $h(\cdot)$ which are one-to-one functions which map onto $(-\pi, \pi)$ satisfying $h(0) = 0$ may be used [28].

5.2 Theory of Random Walks and Estimation

In this section, theory about random walks, Markov processes and their estimation is presented.

5.2.1 Random Walk

The simplest examples of a stochastic process is the white noise process, which is defined as the independent sequence of random variables $X_0, X_1, ...$ which are all identically distributed. A random walk is a stochastic process which is formed by the consecutive summation of identically, independently distributed random variables. Let $\{X_t\colon\ t = 0, 1, 2, ...\}$ denote the white noise process, and let $Y_0, Y_1, ...$ be a sequence of random variables where $Y_0 = 0$ and $Y_t = \sum_{i=1}^{t} X_i$ for all $t \geq 1$. Then, the process $\{Y_t\colon\ t = 0, 1, 2..\}$ is called a random walk process [41].

The random walk has increments which are independent, therefore having the Markov property which indicates that the probabilities of future events depend only on the state of the random walk at the latest point (t_n) in time for which we have information and not on any previous points in time.

A special case of the random walk is the simple random walk. This is the case when X_t is either $+1$ or -1 with respective probabilities p and $1 - p$. This represents independent steps which are to the left with a probability of $1 - p$ or to the right with a probability of p. In this case, Y_t will denote the position of the process after t steps.

5.2.2 Multi-State Random Walk

Another variant of random walks is the multi-state random walk. The primary idea of the multi-state random walk (MRW) is a random walk on a lattice structure, where the process may be at each site in a number of internal states with properties which affect the random walk [7]. Consider random walks in $\mathbb{Z}^d$ with a bounded, symmetrical increment distribution. An irreducibility criterion is imposed to ensure that all points may be reached

in the lattice $\mathbb{Z}^d$ [26]. The integer lattice $\mathbb{Z}^d$ is defined as follows [26]:

$$\mathbb{Z}^d = \{(x^1, ..., x^d) : x^j \in \mathbb{Z}\} \tag{69}$$

where the superscripts denote components and subscripts would denote elements i.e. the point x_j written in component form is $x_j = (x_j^1, ..., x_j^d)$.

The standard basis of unit vectors in $\mathbb{Z}^d$ are denoted by $\mathbf{e}_1 = (1, 0, ..., 0), ..., \mathbf{e}_d = (0, ..., 0, 1)$. Th e prime example in the discrete case would be to consider the simple random walk starting at $k \in \mathbb{Z}^d$. In the case, the simple random walk could be a Markov chain with state space $\mathbb{Z}^d$ and corresponding transition probabilities defined as [26]:

$$P\{S_{n+1} = z | S_n = y\} = \frac{1}{2d}, \qquad z - y \in \{\pm\mathbf{e}_1, ..., \mathbf{e}_d\} \tag{70}$$

Alternatively, this simple random walk may be considered as the sum of a sequence of i.i.d random variables [26]:

$$S_n = k + X_1 + ... + X_n \tag{71}$$

where $P\{X_j = \mathbf{e}_j\} = P\{X_j = -\mathbf{e}_j\} = \frac{1}{2d}$, $j = 1, ..., d$.

5.2.3 Biased Correlated Random Walk / Correlated Random Walk

Random walks were initially used as simple models for movement and were uncorrelated and unbiased. Contextually, uncorrelated indicates that the movement is independent of previous directions, implying that the locations after each successive step in the random walk is dependent only on the location of the previous step, which is indeed the Markov property. The unbiased aspect means that no single direction is preferred, and the direction in which the animal might move at each step is completely random.

Correlated random walks (CRWs) involve a quantified correlation between successive steps called persistence [9]. This 'persistence' produces a local directional bias where each step has a tendency to point in the same direction as the previous step. However, the strength of the local directional bias which diminishes over time. Step orientations are uniformly distributed in the long term [9]. Taking into consideration that animals tend to have a forward persistence to move or walk in a forward direction, CRW's are popularly used to model animal paths in a variety of contexts. By increasing the probability of moving in a certain direction, a global directional bias may be introduced into the model. Paths that contain a consistent bias in a preferred direction or towards a specific target are biased random walks (BRWs) or biased and correlated random walks (BCRWs) in the event that persistence is also observed. This bias may be caused by a multitude of factors ranging from fixed external environmental factors to choices of direction by individuals at each step in the process. The target direction and

bias are not necessarily fixed over the entire path taken and could possibly vary per location and time. Direction, functional form and magnitude of bias may be quantified in the case where directional targets are fixed for all individuals in the population [9].

The primary difference between a CRW and BCRW is that the bias present in a BCRW is consistent and does not diminish over time, whereas the local directional bias in a CRW does diminish over time. CRWs and BCRWs are often referred to as velocity jump processes since both processes involve random changes in velocity over time [9]. BCRWs discretise the path taken and the movement made between times t and $t+1$ is a two-dimensional step vector for all times $t = 0, 1, \ldots, T$. A polar representation (y_t, d_t) of the step vector often used where the step angle $y_t \in [-\pi, \pi)$ provides the direction of the movement occurring at time t with respect to reference direction, where d_t is the respective step length. Let ψ_t be an angle indicating the direction toward the target at time t. This angle ψ_t is the angle of a line joining the animal's position at time t and the target's position with respect to the reference direction [11]. The step angle denoted by y_t is the result of compromising between movement and directional persistence in response to ψ_t. A compromise made by the step angle y_t may be modelled and quantified by setting the mean direction μ_t to the vector direction of the vector [11]

$$\mathbf{v_t} = \begin{pmatrix} \cos(y_{t-1}) \\ \sin(y_{t-1}) \end{pmatrix} + \beta \begin{pmatrix} \cos(\psi_t) \\ \sin(\psi_t) \end{pmatrix}. \tag{72}$$

To find an expression for the mean direction μ_t, a rotation of a de-centred angle which is defined as the direction of the sum of unit vectors which correspond to the angle ψ_t. Suppose the parametrisation for the de-centring vector is $(u_1, u_2) = r(\cos(\alpha), \sin(\alpha))$ where r is a real number and α is an angle [38]. Using this notation, the direction of $(u_1 + \cos(x), u_2 + \sin(x))^T$ may be written as [38]:

$$\begin{aligned} =& \frac{1}{\{r^2 + 1 + 2r.\cos(\alpha - x)\}^{\frac{1}{2}}} \begin{pmatrix} r.\cos(\alpha) + \cos(x) \\ r.\sin(\alpha) + \sin(x) \end{pmatrix} \\ =& \frac{1}{\{r^2 + 1 + 2r.\cos(x - \alpha)\}^{\frac{1}{2}}} \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} r + \cos(x - \alpha) \\ \sin(x - \alpha) \end{pmatrix} \end{aligned}$$

Now, rotating the vector above yields the general form of the de-centred predicted direction as

$$\frac{1}{\{r^2 + 1 + 2r.\cos(x - \alpha)\}^{\frac{1}{2}}} \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} r + \cos(x - \alpha) \\ \sin(x - \alpha) \end{pmatrix}$$

where $\phi \in [0, 2\pi)$. Rewriting this general form in terms of angles will yield the expression for the angle of the de-centred predictor [38]:

$$\begin{aligned}\theta(x, \phi, y_\alpha, r) =& \phi + \text{atan}\{\sin(x - \alpha)\}(\text{mod}2\pi) \\ =& \text{atan}\{r\sin(\phi) + \sin(x + \phi - \alpha), r\cos(\phi) + \cos(x + \phi - \alpha)\}(\text{mod}2\pi),\end{aligned}$$

where

$$\text{atan}(a, b) = \begin{cases} \text{atan}(\frac{a}{b}) \text{ for } b > 0 \\ \text{atan}(\frac{a}{b}) + \pi \text{ for } b < 0. \end{cases}$$

Making use of this parametrisation, the mean μ_t is defined as [11]

$$\mu_t = \text{atan}\{\sin(y_{t-1}) + \beta\sin(\psi_t), \cos(\psi_t) + \beta\cos(\psi_t)\} \tag{73}$$

The parameter β is a weight of the strength of attraction of the target in relation to the directional persistence in the animal's compromise when choosing the following movement angle [11]. A CRW is obtained when $\beta = 0$, resulting in $\mu_t = y_{t-1}$ and a biased random walk (BRW) is obtained when $\beta \to \infty$ resulting in $\mu_t = \psi_{t-1}$. Duchesne et al. (2015) demonstrated that the aforementioned expected movement in the BCRW is defined by (72) since this specific definition of the mean direction provides a stable definition invariant to the addition or subtraction of 2π to the angle.

The estimation of β is of particular importance since it weighs the strength of attraction of the target relative to the animal's directional persistence compromise when choosing the next movement angle. Maximum likelihood methods may be used, but an angular (directional) or consensus model needs to firstly be initialised.

The directional model is obtained by assuming that the step angles y_t are von Mises distributed with average direction of μ_t and a concentration parameter $\kappa > 0$ with pdf (1). This estimation method consists of the parameters β and κ via maximisation of the likelihood constructed with (1) [11].

5.2.4 Markov Process

The Markov process $\{X_t\}$ is a stochastic process with property that when the value of $\{X_t\}$ is given, the values of X_s for $s > t$ are independent the values of X_u for $u < t$ [35]. This implies that the probability of the future behaviour of the process with the current state known, is independent of knowledge of the past behaviour of the process. A stochastic process has the Markov property if the following condition holds [35]:

$$P\{X_{n+1} = j | X_0 = a_0, ..., X_{n-1} = a_{n-1}, X_n = a\}$$

$$=P\{X_{n+1}=j|X_n=a\} \tag{74}$$

for all time points n and states $a_0, ..., a_{n-1}, a, j$.

A discrete Markov chain is classified as a Markov process with a discrete state space and parameter space $T=\{0,1,2,\dots\}$. A stochastic process is called a Markov chain if for all time points n, if it is true that [35]

$$\begin{aligned} P\{X_{n+1}=&a_{n+1}|X_1=a_1,...,X_n=a_n\} \\ =&P\{X_{n+1}=a_{n+1}|X_n=a_n\} \end{aligned} \tag{75}$$

for all states $a_1, ..., a_n, a_{n+1}$. The one-step transition probability denoting the probability of X_{n+1} being in state k given that X_n is in state j and is denoted $P_{jk}^{n,n+1}$. It is defined as,

$$P_{jk}^{n,n+1}=P\{X_{n+1}=k|X_n=j\} \tag{76}$$

If the transition probabilities are independent of n i.e., the time variable, the Markov chain has stationary transition probabilities.

A Markov process is called a Markov jump process if it has a continuous parameter space $T=[0,\infty)$ and a discrete state space. A stochastic process is called a Markov jump process if for all time points n it is true that

$$\begin{aligned} P\{X_{n+1}=&a_{n+1}|X_1=a_1,...,X_n=a_n\} \\ =&P\{X_{n+1}=a_{n+1}|X_n=a_n\} \end{aligned} \tag{77}$$

The transition probability denoting the probability of X_t being in state j given that X_s is in state a is $P\{X_t=j|X_s=a\}$. The transition probability is defined as [35]:

$$P\{X_t=j|X_s=a\}=P_{aj}(s,t). \tag{78}$$

5.2.5 Markov Renewal Process

A renewal event is defined as the event where an object is replaced by another object from the same set the moment it fails [19]. For example, suppose the experiment involves a set of machinery whose lifetimes are independent. This particular experiment consists of using one machine at a time, and when this machine fails, it is immediately replaced. This example constitutes a renewal event. Let $N(t)$ denote the number of renewal events till time t, with the assumption that first machine was turned on at time $t=0$. The time until failure of T_n of the first n

machines is then given by [19]

$$T_0 = 0 \tag{79}$$

$$T_1 = X_1$$

$$T_2 = X_1 + X_2$$

$$\vdots$$

$$T_n = X_1 + X_2 + \cdots + X_n \tag{80}$$

The process $\{N(t), t \geq 0\}$ is therefore a counting process, which is known as a renewal process which is generated by the inter-arrival times $\{T_n, n \geq 1\}$ which denotes the number of renewals up to time t. In a renewal process, the holding times do not necessarily need to be exponentially distributed, and may assume any distribution on positive numbers with the condition that the holding times are independent and identically distributed (i.i.d) with a mean that is finite.

The Markov renewal process is a generalisation of a renewal process where the sequence of holding times is not i.i.d. The distribution of the sequence of holding times are dependent on the states in a Markov chain. Let $\mathbf{S}$ denote the state space and the stochastic process $X_t \in S$ be a Markov chain with $n = 0, 1, 2...$ as the states of the process [41]. Let (X_n, T_n) denote a sequence of random variables, where T_n represent the jump times of the states and X_n are the historical states. The inter-arrival times of the states in T_n are $\tau_n = T_n - T_{n-1}$. The sequence (X_n, T_n) is called a Markov renewal process if for any $n \geq 0$, $\tau \geq 0$,$a, j \in \mathbf{S}$ it is true that [41]:

$$\begin{aligned} P\{\tau_{n+1} \leq \tau, X_{n+1} = j | (X_0, T_0), (X_1, T_1), ..., (X_n = a, T_n)\} \\ = P\{\tau_{n+1} \leq \tau, X_{n+1} = j | X_n = a\}. \end{aligned} \tag{81}$$

The state transition probabilities are defined by [41]:

$$H_{aj}(\tau) = P\{\tau_{n+1} \leq \tau, X_{n+1} = j | X_n = a\}. \tag{82}$$

5.2.6 Hidden Markov Model

A hidden Markov model (HMM) is a Markov chain where the state is exclusively partially observable. This broadly implies that a HMM is a Markov process split into an observable component and an unobservable component. A HMM is characterised by the following [37]:

1. N, which denotes the number of states within the model. The states are generally interconnected so that

any state may be reached from any other state. The state space is denoted by S. The state at a time t is denoted by q_t, and the set of N individual states are denoted as $S = \{S_1, S_2, ..., S_N\}$ [37].

2. M, which represents the number of distinct observational symbols per state corresponding to the physical output of the system being modelled. The individual symbols are $V = \{v_1, ..., v_M\}$.

3. A, which denotes the transition probability matrix, where each a_{ij} denotes the probability of moving from state i to state j such that $\sum_{j=1}^{N} a_{ij} = 1 \ \forall i$. It is defined as [37]

$$a_{ij} = P\{q_{t+1} = S_j | q_t = S_i\}, \quad 1 \leq i, j \leq N. \tag{83}$$

   In the case where each state can reach any other state, $a_{ij} > 0, \forall i, j$, otherwise, $a_{ij} = 0$ for more than one i, j pair.

4. B, which is the probability distribution of the observation symbol in state j, where $B = \{b_j(k)\}$ where

$$b_j(k) = P\{v_k \text{ at } t | q_t = S_j\}, \quad 1 \leqslant j \leqslant N, 1 \leqslant k \leqslant M. \tag{84}$$

5. π, which is the initial state distribution where

$$\pi = \pi_i = P\{q_i = S_i\}, \quad 1 \leqslant i \leqslant N. \tag{85}$$

   Note that π_i is the probability that the Markov chain starts in state i and $\sum_{i=1}^{N} \pi_i = 1$.

The entire parameter space of the HMM is then denoted by $\lambda = (A, B, \pi)$ for simplicity. An ergodic model is a model with the property that each state can reach every other state in a finite number of steps. The figure below illustrates an HMM model with one discrete hidden node and one continuous or discrete node per slice [41].

In Figure 18, the circles represents continuous nodes, squares represent discrete nodes, clear nodes indicate hidden states, and shaded nodes mean that they are observed. An instance of the HMM process is shown in the *trellis*, where thick lines represent state paths and thin lines represent available transitions of states. The state transition probabilities are specified within matrix A i.e., the transition probability matrix. The initial state is s_0 which is chosen according to the initial state distribution π. Equivalently, the underlying Markov chain of the HMM can be expressed by the state transition diagram. The process produces observation o_1 with corresponding emission probability $b_{s_0,s_1} o_1$ while transitioning from state s_0 to s_1, which continues until the final observation o_T

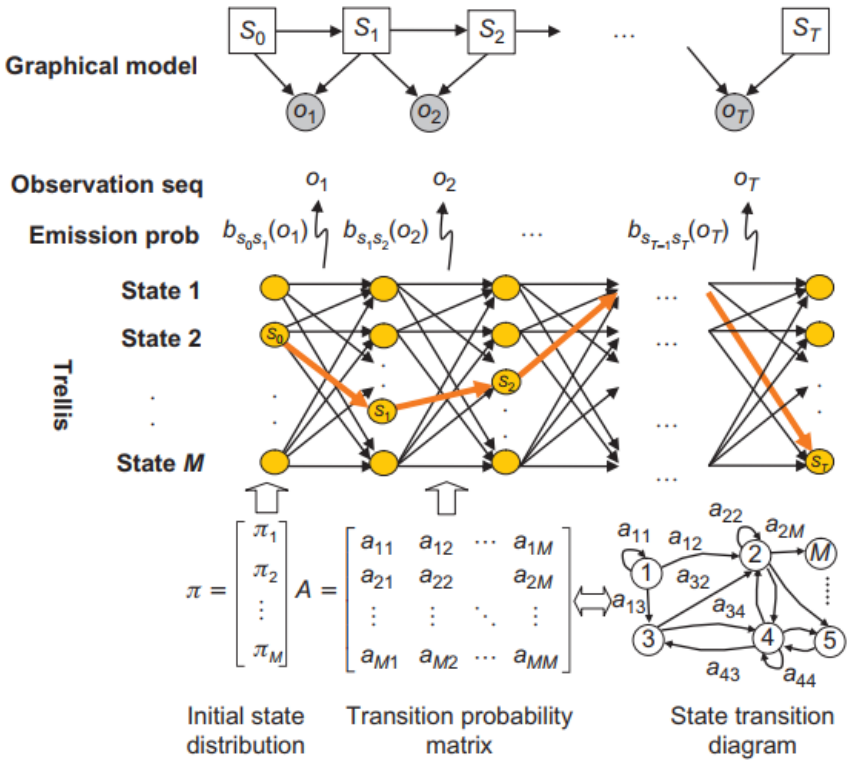


Figure 18: HMM process illustration

5.3 Useful Results

All useful results used for theoretical results are shown below:

Result 1. *The modified Bessel function function of the first kind and of order p is defined by [28]:*

$$I_p(\kappa) = \frac{1}{2\pi}\int_0^{2\pi} \cos(p\theta)\exp(\kappa\cos(\theta))d\theta \tag{86}$$

with a corresponding power series expansion

$$I_p(\kappa) = \sum_{n=0}^{\infty} \frac{1}{\Gamma(p+n+1)\Gamma(n+1)}\left(\frac{\kappa}{2}\right)^{2n+p}.$$

Result 2. *The p^{th} trigonometric moment about the mean direction is given by [28]:*

$$\varphi_p = \bar{a}_p + i\bar{b}_p, \tag{87}$$

where

$$\bar{a}_p = \frac{1}{n}\sum_{i=1}^{n}\cos(\theta_i - \bar{\theta}), \qquad \bar{b}_p = \frac{1}{n}\sum_{i=1}^{n}\sin(\theta_i - \bar{\theta})$$

Then,

$$\varphi_1 = \bar{R}$$

where $\bar{\theta}$ and $\bar{R}$ denote the mean direction and mean resultant length respectively. The mean direction $\bar{\theta}$ is the average of the angles $\theta_1, \ldots, \theta_n$ of the unit vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$. The mean resultant length $\bar{R}$ is given by

$$\bar{R} = \left(\left(\frac{1}{2} \sum_{i=1}^{n} \cos(\theta_i) \right)^2 + \left(\frac{1}{2} \sum_{i=1}^{n} \sin(\theta_i) \right)^2 \right)^{\frac{1}{2}}$$

Result 3. *The binomial expansion states that for $n \in \mathbb{R}$ if $|\frac{a}{b}| < 1$ [3],*

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

where each $\binom{n}{k}$ is a positive integer known as the binomial coefficient.

Result 4. *A continuous random variable X has the exponential distribution with parameter $\theta > 0$ if it has the pdf of the form [3]*

$$f(x;\theta) = \frac{1}{\theta} \exp\left(-\frac{x}{\theta}\right), \quad x > 0 \tag{88}$$

Result 5. *A random variable X has the generalised Gamma distribution $GG(\alpha, \tau, \lambda)$ if it has the pdf of the form [21]:*

$$f(x|\alpha,\tau,\lambda) = \frac{\tau}{\lambda\Gamma(\alpha)} \left(\frac{x}{\lambda}\right)^{(\alpha\tau-1)\left(\exp\left(-\left(\frac{x}{\lambda}\right)\right)^{\tau}\right)}, \quad x \geq 0, \tau, \alpha, \lambda > 0 \tag{89}$$

where $\Gamma(\cdot)$ is the gamma function, α and τ are the shape parameters and λ is the scale parameter.

Result 6. *The Jacobi-Anger expansion is denoted by [2]*

$$\exp(kcos(\theta)) = I_0(k) + 2\sum_{j=1}^{\infty} I_j(k)\cos(j\theta) \tag{90}$$

where $I_j(k)$ is the Bessel function of order j.

Result 7. *The gamma function is denoted by $\Gamma(\kappa)$ for all $\kappa > 0$ is given by [3]*

$$\Gamma(\kappa) = \int_0^{\infty} x^{\kappa-1} \exp(-x) dx. \tag{91}$$

The gamma function also has the following properties:

$$\Gamma(\kappa) = (\kappa-1)\Gamma(\kappa-1) \text{ for } \kappa > 1$$

$$\begin{aligned}\Gamma(n) =&(n-1)!\\ \Gamma\left(\frac{1}{2}\right) =&\sqrt{\pi}\end{aligned} \tag{92}$$

Result 8. *The Euler-Mascheroni constant is defined by [23]:*

$$\gamma = \lim_{n\to\infty}\left(\sum_{i=1}^{n}\frac{1}{i} - \log(n)\right) \tag{93}$$

which approximately equates to $\gamma = 0.57721567$.

Result 9. *There are three important properties regarding the Gamma function, namely [17]:*

1. $\frac{\partial \log\Gamma(x)}{\partial x} = \frac{\Gamma'(x)}{\Gamma(x)} = \Psi(x)$ *where* $\Psi(x) = -\gamma + \sum_{n=0}^{\infty}\left(\frac{1}{n+1} - \frac{1}{x+n}\right)$ *where* γ *is from* (93).
2. $\Gamma'(x) = \Gamma(x)\Psi(x)$.
3. $\Gamma''(x) = \Gamma'(x)\Psi(x) + \Gamma(x)\Psi'(x)$ *where* $\Psi'(x) = \sum_{n=0}^{\infty}\frac{1}{(x+n)^2}$.

Result 10. *The following trigonometric relations and inequalities hold for* $x \in \mathbb{R}$ *[34];*

1. $|\sin(x)| < x$ *and* $|\cos(x)| < 1$
2. $\sin(x) + \cos(y) = 2\sin\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right)$.

6 Appendix B

6.1 Code

All the models have been included in a single code chunk for the reader. The data is available here https://www.sciencedirect.com/science/article/abs/pii/S0167947316301736.

```
library(LindleyR)
library(rmutil)

# This is the Euclidean distance between two points. The parameters x0, y0 are the
    ↪ coordinates of one point. x1, y1 coordinates of the other point.
getDistance <- function(x0, y0, x1, y1) { ##this gets me the Eucledian distance
as.vector(sqrt((x1 - x0)^2 + (y1 - y0)^2))
}

# This is the Angle between two points. It calculates the angle between a vector pointing
    ↪  North whose origin is the point (x0, y0) and a vector whose origin is also point (
    ↪ x0, y0) and which contains point (x1, y1). Input in radians.
getAngle <- function(x0, y0, x1, y1) { ##this gets me the angle between two points
at <- atan2(y1 - y0, x1 - x0)
as.vector(at)
}

# This is the weighted von Mises distribution in the state k. This Function computes the
    ↪ weigthed von Mises distribution in the state k wrt to smooth probabilities.
#kappa is the matrix of parameters associated to the von Mises distributions
# E.smooth: matrix T*K of smooth probabilities
# data: data frame of the directions (y), distances (d), explanatory angles variables (x)
    ↪  and explanatory real variables (z)
#L: the weighted log-likelihood function related to the direction

LL.vonMises <- function(kappa, E.smooth, data, k) { ##Weighted von mises distribution in
    ↪ state k
```

```
        x <- data$x
        y <- data$y
        z <- data$z
        steps <- length(y)
        y0 <- y[1] #inital direction
        L <- 0 #initial LL set to zero
        t <- 1 #time 1

        conc <- sqrt( (kappa %*% (c(sin(y0), z[t, ] * sin(x[t, ])))^2 + (kappa %*% (c(cos
            ↪ (y0), z[t, ] * cos(x[t, ])))^2)

        l.t0 <- E.smooth[t, k] * (-log(2 * pi * besselI(conc, 0, expon.scaled = FALSE)) +
            ↪ kappa %*% (c(cos(y[t] - y0), z[t, ] * cos(y[t] - x[t, ]))))
        if (!is.na(l.t0))
                L <- L + l.t0 #we are updating
                        for (t in (2:(steps))) {
                                conc <- sqrt((kappa %*% (c(sin(y[t - 1]), z[t, ] * sin(x[t,
                                    ↪ ])))^2 + (kappa %*% (c(cos(y[t - 1]), z[t, ] * cos(x[
                                    ↪ t,])))^2)
                                #likelihood
                                l.tk <- E.smooth[t, k] * (-log(2 * pi * besselI(conc,
                                0, expon.scaled = FALSE)) + kappa %*% (c(cos(y[t] -
                                y[t - 1]), z[t, ] * cos(y[t] - x[t, ])))
                                if (!is.na(l.tk))
                                L <- L + l.tk #updating
                        }
                        return(L)
                }
# Weigthed distribution of distance: Function that computes the weigthed distribution of
    ↪ the distances
# During the M step of the EM algorithm, we have to maximize the weigthed distribution of
    ↪  distances.
```

```
#param : parameters c(log(shape),log(scale)) of the distances
#d: vector of observed distances
#weight: vector of weight of each observation
# dist: type of distribution on the step length. The choices are 'gamma','weibull','
    ↪ lindley' and 'gumbel'.
#L: the weighted distribution related to the distance
l.dist = function(param, d, weight, dist = "gumbel") {
       shape = exp(param[1])
       scale = exp(param[2])
       if (dist == "gamma")
       L = sum(weight * dgamma(d, shape = shape, scale = scale, log = TRUE))
              if (dist == "weibull")
              L = sum(weight * dweibull(d, shape = shape, scale = scale, log = TRUE))
              if(dist == "lindley") #this is new. Trying things
              L = sum(weight * dplindley(d,shape, abs(scale),log=TRUE))
              if(dist == "fisk")
              L = sum(weight * dfisk(d, scale = scale,shape1.a = shape, log = TRUE))
              if(dist == "lomax")
              L = sum(weight* dlomax(d,scale,shape,log = TRUE))
              if(dist == "gumbel")
              L = sum(weight* dgumbel(d,scale ,shape,log = TRUE))
              return(L)
       }

       # Filtering-Smoothing algorithm : computes filtering and smoothing probabilities
       #param: list of parameters of the model
       #data: data frame of the directions (y), distances (d), explanatory angles
           ↪ variables (x) and explanatory real variables (z)
       #type: angle-dist' for bivariate model on direction-step length and 'angle' for
           ↪ univariate model on direction
       #dist: type of distribution on the step length. The choices are 'gamma','weibull
           ↪ ','lindley' and 'gumbel'.
```

```
#E.smooth: Matrix T*K of the smooth probabilities {P(S_{tk}=1|F_{T})}
#E.filter: Matrix T*K of the filter probabilities {P(S_{tk}=1|F_{t-1})}
#LogLike: Value of the log-likelihood function w.r.t the parameters
FilterSmooth <- function(param, data, type = "angle", dist = "gumbel") {
       # initialization:
       pi0 <- param$pi0 #starting values of 0.7 and 0.3 for initial pi
       P <- param$P #the markov chain(the transition matrix)
       kappa <- param$kappa
       thetalin <- param$thetalin
       K <- length(pi0) #2, since we are doing a 2 state HMM
       p <- length(kappa[1, ]) - 1
       # information on the data set
       x <- as.matrix(data$x)
       y <- as.matrix(data$y)
       z <- as.matrix(data$z)
       d <- as.matrix(data$d)
       steps <- length(y)
       y0 <- y[1]
       E.smooth0 <- rep(0, K)
       E.filter <- matrix(0, steps, K)
       LogLike <- 0 #starting point
       i <- 1
       P.trans <- as.vector(pi0 %*% P) #updating the transition probability matrix
       #likelihood
       l <- as.vector(sqrt((kappa %*% (c(sin(y0), z[i, ] * sin(x[i, ])))^2 + (
           ↪ kappa %*% (c(cos(y0), z[i, ] * cos(x[i, ])))^2))
       #von mises density
       f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE)))) * exp(
           ↪ kappa %*% (c(cos(y[i] - y0), z[i, ] * cos(y[i] -x[i, ])))))
       g <- rep(1, length(f)) #We will store the density of the distance
       if (type == "angle-dist" & dist == "gamma")
       g <- dgamma(d[1], shape = thetalin[, 1], scale = thetalin[, 2])
```

```
if (type == "angle-dist" & dist == "weibull")
g <- dweibull(d[1], shape = thetalin[, 1], scale = thetalin[, 2])
if (type == "angle-dist" & dist == "lindley")
g <- dplindley(d[1], abs(thetalin[, 1]), abs(thetalin[,2]))
if (type == "angle-dist" & dist == "gumbel")
g <- dgumbel(d[1], thetalin[, 1], thetalin[,2])

f.renorm <- as.vector(exp(log(f * g) - max(log(f * g))))
denom <- (P.trans) %*% f.renorm
E.filter[1, ] <- as.vector((P.trans * f.renorm))/denom #the E step

if (!is.na(log((P.trans) %*% (f * g))))
LogLike <- log((P.trans) %*% (f * g)) #updating loglikelihood

# 1. filtering Algorithm
for (i in (2:steps)) {

       P.trans <- as.vector(E.filter[i - 1, ] %*% P) #updating transition
           ↪ probability matrix
       l <- as.vector(sqrt((kappa %*% (c(sin(y[i - 1]), z[i, ] * sin(x[i,
           ↪ ]))))^2 + (kappa %*% (c(cos(y[i - 1]), z[i, ] * cos(x[i, ])))
           ↪ )^2))
       f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE))))
           ↪ * exp(kappa %*% (c(cos(y[i] - y[i - 1]), z[i, ] * cos(y[i] -
           ↪ x[i, ])))))) #von mises pdf
       g <- rep(1, length(f)) #density of the distance distribution
       #we have the weibull option and the gamma option
       #The gamma was preferred based on AIC/BIC
       if (type == "angle-dist" & dist == "gamma")
       g <- dgamma(d[i], shape = thetalin[, 1], scale = thetalin[, 2])
       if (type == "angle-dist" & dist == "weibull")
       g <- dweibull(d[i], shape = thetalin[, 1], scale = thetalin[,2])
```

```
        if (type == "angle-dist" & dist == "lindley")
        g <- dplindley(d[1], abs(thetalin[, 1]), abs(thetalin[,2]))
        if (type == "angle-dist" & dist == "gumbel")
        g <- dgumbel(d[1], thetalin[, 1], thetalin[,2])
        f.renorm <- as.vector(exp(log(f * g) - max(log(f * g)))) #the "
            ↪ difference"
        denom <- (P.trans) %*% f.renorm
        E.filter[i, ] <- as.vector(P.trans * f.renorm)/denom
        if (!is.na(log((P.trans) %*% (f * g))))
        LogLike <- LogLike + log((P.trans) %*% (f * g))
}

# 2. Smoothing Algorithm
E.smooth <- matrix(0, steps, K)
E.smooth[steps, ] <- E.filter[steps, ]
for (t in (steps - 1):1) {
        for (l in (1:K)) {
                for (k in (1:K)) {
                        denom <- (E.filter[t, ]) %*% P[, k]
                        num <- (P[l, k] * E.filter[t, l] * E.smooth[t + 1, k
                            ↪ ])
                        if (!is.na(num/denom)) #ensuring we do not get null
                            ↪ results with an iteration
                        E.smooth[t, l] <- E.smooth[t, l] + num/denom
                }
        }
}
for (l in (1:K)) {
        for (k in (1:K)) {
                denom0 <- pi0 %*% P[, k]
                num0 <- P[l, k] * pi0[l] * E.smooth[1, k]
                if (!is.na(num0/denom0))
```

```
                        E.smooth0[l] <- E.smooth0[l] + num0/denom0
                }
        }
        out <- list(E.smooth0 = E.smooth0, E.filter = E.filter, E.smooth = E.smooth
            ↪ , LogLike = LogLike)
        class(out) = "Filter-Smooth" #assigning the class using brute force to
            ↪ ensure we know what is happening in the main code script
        out
}



#Observed Log-likelihood function with a Markovian hidden process: compute the
    ↪ observed log-likelihood given the assumption that the underlying hidden
    ↪ process is Markovian.
#theta: vector of all the parameters of the model in this order: P, kappa and
    ↪ thetalin
#data: data frame of the directions (y), distances (d), explanatory angles
    ↪ variables (x) and explanatory real variables (z)
#pi0: vector of initial distribution of the hidden process
#nb.target: number of targets
#type of the model: 'angle-dist' for bivariate model on direction-step length and
    ↪ 'angle' for univariate model on direction (Default).
#dist: type of distribution on the step length. The choices are 'gamma' (Default)
    ↪ or 'weibull'

logL <- function(theta, data, pi0, nb.target, type, dist) {
        x <- data$x
        y <- data$y
        z <- data$z
        d <- data$d

        steps <- length(y)
```

```
y0 <- y[1]

p <- nb.target #number of targets
K <- length(pi0) #number of states

kappa <- NULL
P <- NULL

for (l in (1:K)) {
        #note: L is the increment in the loop, not a parameter!!
        #K is the number of states in the HMM (2 state HMM)
        #theta and kappa are the estimated parameter(vectors) with the
            ↪ relevent degrees of freedom
        # to correctly account for the states in the transition probability
            ↪ matrix
        # the rows will sum to 1! This needs to happen!
        # The condition is acocunted for in the code.

        P <- rbind(P, c(theta[(1 + (l - 1) * (K - 1)):(1 + (l - 1) * (K - 1)
            ↪  + K - 2)], 1 - sum(theta[(1 + (l - 1) *
        (K - 1)):(1 + (l - 1) * (K - 1) + K - 2)])))
        kappa <- rbind(kappa, theta[((K * (K - 1) + 1 + (l - 1) * (p + 1)))
            ↪ :((K * (K - 1) + 1) + (l - 1) * (p +
        1) + (p))])
}

thetalin <- theta[c((length(theta) - K * 2 + 1):length(theta))]
E.filter <- matrix(0, steps, K)
LogLike <- 0 #initial value

x <- as.matrix(x)
z <- as.matrix(z)
```

```
cond <- "OK"
i <- 1
P.trans <- as.vector(pi0 %*% P) #updating the transition probability matrix

l <- as.vector(sqrt((kappa %*% (c(sin(y0), z[i, ] * sin(x[i,])))^2 + (
    ↪ kappa %*% (c(cos(y0), z[i, ] * cos(x[i, ])))^2)) #likelihood

f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE)))) * exp(
    ↪ kappa %*% (c(cos(y[i] - y0), z[i, ] * cos(y[i] - x[i, ])))) #von
    ↪ mises

g <- rep(1, length(f)) #distance distribution/density
if (type == "angle-dist") {
       if (dist == "gamma")
       g <- dgamma(d[1], shape = thetalin[c(1, 3)], scale = thetalin[c(2,4)
           ↪ ])
       if (dist == "weibull")
       g <- dweibull(d[1], shape = thetalin[c(1, 3)], scale = thetalin[c(2,
           ↪  4)])
       if (dist == "lindley")
       g <- dplindley(d[1], abs(thetalin[c(1, 3)]), abs(thetalin[c(2, 4)]))
       if (dist == "gumbel")
       g <- dgumbel(d[1], thetalin[c(1, 3)],thetalin[c(2, 4)])}

f.renorm <- as.vector(exp(log(f * g) - max(log(f * g)))) #Update

denom <- (P.trans) %*% f.renorm
E.filter[1, ] <- as.vector((P.trans * f.renorm))/denom

if (is.na(log((P.trans) %*% (f * g))) == FALSE)
LogLike <- log((P.trans) %*% (f * g))
# 1. Algorithm filtering
```

```
for (i in (2:steps)) {
        P.trans <- as.vector(E.filter[i - 1, ] %*% P)
        l <- as.vector(sqrt((kappa %*% (c(sin(y[i - 1]), z[i,] * sin(x[i, ])
            ↪ )))^2 + (kappa %*% (c(cos(y[i - 1]),z[i, ] * cos(x[i, ])))))
            ↪ ^2))
        f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE))))
            ↪ * exp(kappa %*% (c(cos(y[i] - y[i - 1]), z[i, ] * cos(y[i] -x
            ↪ [i, ])))))
        if (type == "angle-dist") {
                if (dist == "gamma")
                g <- dgamma(d[i], shape = thetalin[c(1, 3)],
                scale = thetalin[c(2, 4)])
                if (dist == "weibull")
                g <- dweibull(d[i], shape = thetalin[c(1, 3)],
                scale = thetalin[c(2, 4)])
                if (dist == "lindley")
                g <- dplindley(d[1], thetalin[c(1, 3)], abs(thetalin[c(2, 4)
                    ↪ ]))
                if (dist == "gumbel")
                g <- dgumbel(d[1], thetalin[c(1, 3)], thetalin[c(2, 4)])

        }

        f.renorm <- as.vector(exp(log(f * g) - max(log(f * g))))
        denom <- (P.trans) %*% f.renorm
        E.filter[i, ] <- as.vector(P.trans * f.renorm)/denom
        if (is.na(log((P.trans) %*% (f * g))) == FALSE)
        LogLike <- LogLike + log((P.trans) %*% (f * g))
}

return(LogLike) #this likelihood will tell us whether or not the underlying
    ↪  process is markovian or
```

```
}



#Transition matrix is semi_Markov process:Function that compute the transition
    ↪ matrix of the approximated semi-Markov process
# m: vector of the size of the Dwell times
#p: vector of probabilities of the Dwell times.
#Gamma: Transition matrix

gen.Gamma <- function(m, p) {
        Gamma <- diag(sum(m)) * 0
        ## state aggregate 1
        if (m[1] == 1) {
                #gamma = transition probability matrix = (1-q1, q1, q2,1-q2) 2x2
                Gamma[1, 1] <- 1 - dnbinom(0, size = p[1], prob = p[3])
                Gamma[1, 2] <- 1 - Gamma[1, 1]
        }
        if (m[1] > 1) {
                Gamma[1, m[1] + 1] <- dnbinom(0, size = p[1], prob = p[3])
                Gamma[1, 2] <- 1 - Gamma[1, m[1] + 1]
                for (i in 2:(m[1] - 1)) {
                        cc <- rep(1, sum(m))
                        for (k in 1:(i - 1)) {
                                cc[k] <- Gamma[k, k + 1]
                        }
                        dd <- prod(cc)
                        if (dd > 1e-12)
                        Gamma[i, m[1] + 1] <- dnbinom(i - 1, size = p[1],
                        prob = p[3])/dd
                        if (dd < 1e-12)
                        Gamma[i, m[1] + 1] <- 1
```

```
                Gamma[i, i + 1] <- 1 - Gamma[i, m[1] + 1]
        }
        cc <- rep(1, sum(m))
        for (k in 1:(m[1] - 1)) {
                cc[k] <- Gamma[k, k + 1]
        }
        dd <- prod(cc)
        if (dd > 1e-12)
        Gamma[m[1], m[1] + 1] <- dnbinom(m[1] - 1, size = p[1],
        prob = p[3])/dd
        if (dd < 1e-12)
        Gamma[m[1], m[1] + 1] <- 1
        Gamma[m[1], m[1]] <- 1 - Gamma[m[1], m[1] + 1]
}
## state aggregate 2
if (m[2] == 1) {
        Gamma[2, 2] <- 1 - dnbinom(0, size = p[2], prob = p[4])
        Gamma[2, 1] <- 1 - Gamma[2, 2]
}
if (m[2] > 1) {
        Gamma[m[1] + 1, 1] <- dnbinom(0, size = p[2], prob = p[4])
        Gamma[m[1] + 1, m[1] + 2] <- 1 - Gamma[m[1] + 1, 1]
        for (i in 2:(m[2] - 1)) {
                cc <- rep(1, sum(m))
                for (k in 1:(i - 1)) {
                        cc[k] <- Gamma[m[1] + k, m[1] + k + 1]
                }
                dd <- prod(cc)
                if (dd > 1e-12)
                Gamma[m[1] + i, 1] <- dnbinom(i - 1, size = p[2],
                prob = p[4])/dd
                if (dd < 1e-12)
```

```
                Gamma[m[1] + i, 1] <- 1
                Gamma[m[1] + i, m[1] + i + 1] <- 1 - Gamma[m[1] + i, 1]
        }
        cc <- rep(1, sum(m))
        for (k in 1:(m[2] - 1)) {
                cc[k] <- Gamma[m[1] + k, m[1] + k + 1]
        }
        dd <- prod(cc)
        if (dd > 1e-12)
        Gamma[m[1] + m[2], 1] <- dnbinom(m[2] - 1, size = p[2],
        prob = p[4])/dd
        if (dd < 1e-12)
        Gamma[m[1] + m[2], 1] <- 1
        Gamma[m[1] + m[2], m[1] + m[2]] <- 1 - Gamma[m[1] + m[2], 1]
    }
    Gamma
}

#Observed Log-likelihood function with a semi-Markovian hidden process: Function
    ↪ that compute the observed log-likelihood given the assumption that the
    ↪ underlying hidden process is semi markov


logLSemi <- function(theta, y0, data, pi0, nb.target, m = c(30,
30), type = "angle-dist", dist = "gumbel") {
    if (m[1] == 1) #if we have accuracy of 1 for state 1
    theta[1] = 1
    if (m[2] == 1)
    theta[3] = 1


    theta[c(1, 3)] <- exp(theta[c(1, 3)])
```

```
theta[c(2, 4)] <- inv.logit(theta[c(2, 4)]) #exp(x)/1-exp(x)
n <- length(theta)
theta[c((n - 3):n)] <- exp(theta[c((n - 3):n)])
p <- nb.target #no. of targets
K <- length(pi0) #length of inital distribution of the hidden process
x <- data$x
y <- data$y
z <- data$z
d <- data$d
steps <- length(y)
p1 <- c(theta[1], theta[2])
p2 <- c(theta[3], theta[4])
kappa <- rbind(theta[5:(5 + (p))], theta[(5 + p + 1):(length(theta) -
4)])
thetalin <- theta[c((length(theta) - 3):length(theta))]

P <- gen.Gamma(m, cbind(c(p1[1], p2[1]), c(p1[2], p2[2]))) #this is the
    ↪ transition matrix of a semi markov process
kappacopy <- NULL
thetalincopy <- NULL
for (i in (1:m[1])) { #1 till 30
       kappacopy <- rbind(kappacopy, kappa[1, ])
       thetalincopy <- rbind(thetalincopy, thetalin[1:2])
}
for (i in (1:m[2])) {
       kappacopy <- rbind(kappacopy, kappa[2, ])
       thetalincopy <- rbind(thetalincopy, thetalin[3:4])
}

kappa <- kappacopy
thetalin <- thetalincopy
param = list(pi0 = pi0, kappa = kappacopy, thetalin = thetalincopy,
```

```
P = P)
data <- list(x = x, y = y, z = z, d = d)
type = "angle-dist"

E.filter <- matrix(0, steps, K)
LogLike <- 0
x <- as.matrix(x)
z <- as.matrix(z)
cond <- "OK"
# initialization
i <- 1
P.trans <- as.vector(pi0 %*% P)
l <- as.vector(sqrt((kappa %*% (c(sin(y0), z[i, ] * sin(x[i,
]))))^2 + (kappa %*% (c(cos(y0), z[i, ] * cos(x[i, ]))))^2))
f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE)))) *
exp(kappa %*% (c(cos(y[i] - y0), z[i, ] * cos(y[i] -
x[i, ])))))
if (type == "angle-dist") {
       if (dist == "gamma")
       g <- dgamma(d[1], shape = thetalin[, 1], scale = thetalin[,
       2])
       if (dist == "weibull")
       g <- dweibull(d[1], shape = thetalin[, 1], scale = thetalin[,
       2])
       if (dist == "lindley")
       g <- dplindley(d[1], abs(thetalin[, 1]), abs(thetalin[, 2]))

       if (dist == "fisk")
       g <- dfisk(d[1], thetalin[, 1], thetalin[, 2])
       if (dist == "lomax")
       g <- dlomax(d[1], thetalin[, 1], thetalin[, 2])
       if (dist == "gumbel")
```

```
        g <- dgumbel(d[1], thetalin[, 1], thetalin[, 2])

}
f.renorm <- as.vector(exp(log(f * g) - max(log(f * g))))
denom <- (P.trans) %*% f.renorm
E.filter[1, ] <- as.vector((P.trans * f.renorm))/denom

if (is.na(log((P.trans) %*% (f * g))) == FALSE)
LogLike <- log((P.trans) %*% (f * g))
# 1. Filtering algorithm

for (i in (2:steps)) {
        P.trans <- as.vector(E.filter[i - 1, ] %*% P)
        l <- as.vector(sqrt((kappa %*% (c(sin(y[i - 1]), z[i,] * sin(x[i, ])
            ↪ )))^2 + (kappa %*% (c(cos(y[i - 1]), z[i, ] * cos(x[i, ])))))
            ↪ ^2))
        f <- as.vector((1/(2 * pi * (besselI(l, 0, expon.scaled = FALSE))))
            ↪ * exp(kappa %*% (c(cos(y[i] - y[i - 1]), z[i, ] * cos(y[i] -
            ↪ x[i, ]))))))
        if (type == "angle-dist") {
                if (dist == "gamma")
                g <- dgamma(d[i], shape = thetalin[, 1], scale = thetalin[,
                2])
                if (dist == "weibull")
                g <- dweibull(d[i], shape = thetalin[, 1], scale = thetalin[,
                2])
                if (dist == "lindley")
                g <- dplindley(d[1], abs(thetalin[, 1]), abs(thetalin[, 2]))

                if (dist == "gumbel")
                g <- dgumbel(d[1], thetalin[, 1], thetalin[, 2])
        }
```

```
            f.renorm <- as.vector(exp(log(f * g) - max(log(f * g))))
            denom <- (P.trans) %*% f.renorm
            E.filter[i, ] <- as.vector(P.trans * f.renorm)/denom
            if (is.na(log((P.trans) %*% (f * g))) == FALSE)
            LogLike <- LogLike + log((P.trans) %*% (f * g))
        }
        return(LogLike)
}



############################## EM-algorithm ###############


# Fit the general hidden random walk model using initial parameter
#param: list of all the parameters of the model in this order: P, kappa and
    ↪ thetalin.
#data: data frame of the directions (y), distances (d), explanatory angles
    ↪ variables (x) and explanatory real variables (z)
#EMMax: numbers of EM algortihm's maximum iteration
#EMMin: numbers of EM algortihm's minimum iteration
#precision: of the convergence, i.e. EM converge if the minimum between two
    ↪ consecutives estimations is less than 10^(-{precision})
#beta: Matrix p*K of normalized coefficient beta associated to the targets.
#LL: Value of the maximized log-likelihood function
#s: Numbers of EM algirithm's iteration
#P:estimated transition matrix
#pi0:initial probability distribution
#kappa: matrix (p+1)*K of the parameters associated to the direction
#thetalin: matrix 2*K of the parameters associated to the distance
#thetathistory: of estimated parameters in the EM algorithm
```

```
fitGHRW <- function(param, data, EMMax = 50, precision = 2, EMMin = 10, type = "
    ↪ angle-dist", dist = "gumbel") {

        # initialization:
        pi0 <- param$pi0 #initial probability distribution
        P <- param$P #initial transition probability matrix
        kappa <- as.matrix(param$kappa) #directional parameters
        thetalin <- param$thetalin #distance parameters
        K <- length(pi0) #no. of states
        p <- length(kappa[1, ]) - 1 # no. of targets

        x <- as.matrix(data$x)
        y <- as.vector(data$y)
        z <- data$z
        d <- data$d
        steps <- length(y)
        y0 <- y[1]

        thetat <- E.smooth <- E.filter <- E.smooth0 <- paramop <- param1 <- param2
            ↪ <- NULL
        LL <- 2 #starting point
        LL.prev <- 1
        s <- 1

        paramop <- param1 <- param2 <- param3 <- NULL

        for (i in seq(along = pi0)) {
                param1 <- c(param1, P[i, -K]) #q1
                param2 <- c(param2, t(kappa[i, ]))
                param3 <- c(param3, t(thetalin[i, ]))
        }
```

```
paramopold <- c(param1, param2, param3)
paramopnew <- paramopold + 1

# Em algorithm

while (norm(as.matrix(paramopnew - paramopold), type = "M") >
10^(-precision) && s < EMMin) {

        paramopold <- paramopnew
        ############ E-step
        if (s > EMMax) { #EMMAx is the max iteration we're willing to do.
                break
        }

        #filtering smoothing algorithm
        FS <- FilterSmooth(param, data, type, dist)
        E.smooth0 <- FS$E.smooth0
        E.filter <- FS$E.filter
        E.smooth <- FS$E.smooth
        LL.prev <- FS$LogLike
        Pa <- param$P

        ############ M-step

        num <- matrix(0, K, K)
        P <- matrix(0, K, K)
        for (h in seq(along = pi0)) {
                denom <- sum(E.smooth[-steps, h]) + E.smooth0[h]
                for (k in seq(along = pi0)) {
                        for (t in c(2:length(y[]))) {
                                v <- E.filter[t - 1, ] %*% Pa
                                num[h, k] <- num[h, k] + E.smooth[t, k] * Pa[h
```

```
                        ↪ ,
                    k] * E.filter[t - 1, h]/v[k]
                }

                v <- pi0 %*% Pa
                num[h, k] <- num[h, k] + E.smooth[1, k] * Pa[h,
                k] * pi0[h]/v[k]
                P[h, k] <- num[h, k]/denom
        }
}

for (i in seq(along = pi0)) {
        op <- optim(param$kappa[i, ], LL.vonMises, E.smooth = E.
            ↪ smooth,
        data = data, k = i, control = list(fnscale = -1),
        method = "L-BFGS-B") #optimises for us!!!!
        kappa[i, ] <- op$par
        if (type == "angle-dist")
        op2 <- optim(log(param$thetalin[i, ]), l.dist,
        d = d, weight = E.smooth[, i], dist = dist,
        control = list(fnscale = -1), method = "L-BFGS-B")
        thetalin[i, ] <- exp(op2$par)
}


ind <- order(-kappa[, 1])
kappa <- kappa[ind, ]
thetalin <- thetalin[ind, ]
P <- P[ind, ind]

param <- list(pi0 = pi0, kappa = kappa, thetalin = thetalin,
P = P)
```

```
        FSL <- FilterSmooth(param, data, type, dist)
        LL <- FSL$LogLike

        # check for spurious maximum
        beta <- kappa[, 2:(p + 1)]/kappa[, 1]
        a <- eigen(t(P))
        ind <- which.max(abs(a$values))
        pstatio <- as.matrix(abs(a$vectors[, ind])/norm(as.matrix(a$vectors
            ↪ [,
        ind]), "o"))
        critkappa <- max(abs(kappa[, 1]))
        # look for a spurious maxima
        if (min(pstatio) < 1e-04 | critkappa > 100 | max(abs(beta)) >
        100 | min(P) < 10^(-5) | min(thetalin) < 0.01 | max(thetalin) >
        20)
        break

        s <- s + 1

        paramop <- param1 <- param2 <- param3 <- NULL

        for (i in seq(along = pi0)) {
                param1 <- c(param1, P[i, -K])
                param2 <- c(param2, t(kappa[i, ]))
                param3 <- c(param3, t(thetalin[i, ]))
        }
        paramop <- c(param1, param2, param3)
        paramopnew <- paramop
        thetat <- rbind(thetat, paramop)
}

FSL <- FilterSmooth(param, data, type, dist)
```

```
        LL <- FSL$LogLike
        out <- list(beta = beta, LL = LL, s = s, P = P, pi0 = pi0,
        kappa = kappa, thetalin = thetalin, thetat = thetat)
        class(out) = "fitGHRW"
        out

}

############################## Global maximum of the likelihood function
    ↪ ###############


# Fit the general hidden random walk model: Function that find the global maximum
    ↪ of the likelihood of General hidden random walk model on data
#EMMax: vectors of size 2 of the numbers of EM algortihm's maximum iteration (
    ↪ short, long)
#EMMin: vectors of size 2 of the numbers of EM algortihm's minmum iteration (short
    ↪ , long)
#precision: vector of size 2 of the convergence of the precision of short and long
    ↪  run algorithm
#semi indicates TRUE for the semi-Markov estimation. Default is FALSE.
#Markov:List of estimation with a Markovian hidden structure
#SemiMarkov: List of estimation with a semi-Markovian hidden structure
#EM.itermax: Number of EM algorithm iterations
#fit: of the model with {likelihood}, {AIC}, {BIC}
#kappa: matrix of the estimated parameters associated to the direction with
    ↪ standard errors.
#phi: matrix of the estimated parameters associated to the distance with standard
    ↪ errors.
#P.Markov:estimated transition matrix with standard errors. Note that for each
    ↪ state k, only k-1 probabilites are displayed, i.e. P(S_t=j|S_{t-1}=k), for
    ↪ j=1...K-1.
```

```
#Dwell.SemiMarkov: estimated parameters of the Dwell time in each state of the
    ↪ semi_Markov process

GHRandomWalk <- function(data, K, paraminitiaux = NULL, nb_init = 50, EMMax = c
    ↪ (50, 2000), precision = c(2, 8), EMMin = c(10, 1000),
type = "angle-dist", dist = "gumbel", semi = FALSE) {

        pi0 <- paraminitiaux$pi0
        kappainit <- paraminitiaux$kappainit
        thetalininit <- paraminitiaux$thetalininit
        L = length(thetalininit[1, ])
        Pinit <- paraminitiaux$Pinit

        x <- as.matrix(data$x)
        y <- as.vector(data$y)
        z <- data$z
        d <- data$d
        steps <- length(y)
        y0 <- y[1]

        # 1 etat
        if (K == 1)
        print("please use the consensus function")
        if (K > 1) {
                # initialization
                LLprec <- -exp(100)
                condition <- FALSE
                thetat1 <- thetat <- NULL
                if (is.null(pi0)) {
                        w <- runif(K)
                        pi0 <- w/sum(w)
                }
```

```
# Estimation for nb_init initial parameters
print("short-run EM algorithm")
j <- 1
while (j < nb_init) {
        if (is.null(kappainit)) {
                kappa <- NULL
                thetalin <- NULL
                for (i in seq(along = pi0)) {
                        conc <- runif(1, min = 2, max = 30) #random
                            ↪ deviates from uniform distribution
                        kappa <- rbind(kappa, c(conc, conc * runif(p,
                            ↪ min = -1, max = 1)))
                }
        }
        if (type == "angle-dist") { #for the bivariate model on
            ↪ direction-step length
                if (is.null(thetalininit))
                thetalin <- matrix(runif(K * L, min = 1, max = 10), K
                    ↪ , L)
        }
        if (!is.null(kappainit))
        kappa <- kappainit + t(replicate(K, runif(p + 1, -1, 1)))

        if (!is.null(thetalininit)) {
                if (type == "angle-dist")
                thetalin <- thetalininit + matrix(runif(K * L, min =
                    ↪ 1, max = 5), K, L)
        }
        if (is.null(Pinit)) {
                P <- NULL
                for (i in seq(along = pi0)) {
                        vect <- runif(K)
```

```
                P <- rbind(P, vect/sum(vect))
        }
}
param <- list(kappa = kappa, thetalin = thetalin,
P = P, pi0 = pi0)

# Estimation
S <- fitGHRW(param, data, EMMax[1], precision[1], EMMin[1],
    ↪ type, dist) #fitting the model
SP <- S$P
a <- eigen(t(P)) #getting the eigenvalues of the transition
    ↪ matrix
LL <- S$LL
ind <- which.max(abs(a$values))
#getting the normalised ratios
pstatio <- as.matrix(abs(a$vectors[, ind])/norm(as.matrix(
    ↪ a$vectors[,ind]), "o"))
critkappa <- min(abs(S$kappa[, 1]))

# check for False maxima
if (min(pstatio) > 1e-04 && critkappa < 100 && max(abs(S$beta
    ↪ )) <
100 && min(P) > 10^(-5) && min(thetalin) > 0.01 &&
LL > LLprec) {
        condition <- TRUE
        kappaop <- S$kappa
        Pop <- S$P
        thetalinop <- S$thetalin
        LLop <- S$LL
        thetat <- S$thetat
        LLprec <- LLop
}
```

```
        print(paste0(round(j/nb_init * 100), "% of the short-run EM
            ↪ algorithm done"))
        j <- j + 1

}
thetat1 <- thetat

# Long-run EM algorithm

if (condition == "FALSE") {
        LL <- -exp(100)
        s <- 0
        print("Not enough initial parameters! Please try again.")
        break
}
if (condition == "TRUE") {
        print("long-run EM algorithm...")
        param <- list(kappa = kappaop, thetalin = thetalinop,
        P = Pop, pi0 = pi0)
        S <- fitGHRW(param, data, EMMax[2], precision[2],
        EMMin[2], type, dist)
        kappaop <- S$kappa
        Pop <- S$P
        thetalinop <- S$thetalin
        beta <- S$beta
        s <- S$s
        LL <- S$LL
        param.Markov <- list(kappa = kappaop, thetalin = thetalinop,
        P = Pop, pi0 = pi0)

        thetat <- rbind(thetat1, S$thetat)
        theta1 <- theta2 <- theta3 <- NULL
```

```
        for (i in seq(along = pi0)) {
                theta1 <- c(theta1, Pop[i, -K])
                theta2 <- c(theta2, (kappaop[i, ]))
                theta3 <- c(theta3, thetalinop[i, ])
        }
        theta <- c(theta1, theta2, theta3)
        theta.Markov = theta
        nb.target <- length(kappaop[1, ]) - 1
}
thetat <- thetat[-1, ]

if (length(thetat) == 0) {
        print("need more initial condition")
        break
}
V3 <- optim(theta, fn = logL, gr = NULL, data = data,
pi0 = param$pi0, nb.target = nb.target, type = type,
dist = dist, method = "L-BFGS-B", control = list(fnscale = -1),
hessian = TRUE, lower = theta - 10^-4 * rep(1, length(theta)),
upper = theta + 10^-4 * rep(1, length(theta)))
if (det(V3$hessian) != 0)
V <- -(solve(V3$hessian))
if (det(V3$hessian) == 0)
V <- -pinv(V3$hessian)

paramkappafinal <- NULL
for (i in seq(along = pi0)) {
        se = sqrt(diag(V[(K * (K - 1) + 1 + (i - 1) * (p +
        1)):(K * (K - 1) + 1 + p + (i - 1) * (p + 1)),
        (K * (K - 1) + 1 + (i - 1) * (p + 1)):(K * (K -
        1) + 1 + p + (i - 1) * (p + 1))]))
        paramkappa <- cbind(kappaop[i, ], se, round(1 - pnorm(abs(
```

```
            ↪ kappaop[i,
        ]/se)), 4))
        colnames(paramkappa) <- c(paste0("estimate ", "(k=",
        i, ")"), paste0("s.e. ", "(k=", i, ")"), paste0("p.value ",
        "(k=", i, ")"))
        paramkappafinal <- cbind(paramkappafinal, paramkappa)
}
paramPfinal <- NULL
for (i in seq(along = pi0)) {
        paramP <- cbind(as.matrix(Pop[i, -K]), as.matrix(sqrt(diag(as
            ↪ .matrix(V[(1 +
        (i - 1) * (K - 1)):(1 + (i - 1) * (K - 1)), (1 +
        (i - 1) * (K - 1)):(1 + (i - 1) * (K - 1))])))))
        colnames(paramP) <- c(paste0("estimate ", "(P", i,
        ".)"), paste0("s.e. ", "(P", i, ".)"))
        paramPfinal <- cbind(paramPfinal, paramP)
}
rownames(paramPfinal) <- paste0("P.", 1:(K - 1))

paramdistfinal <- NULL
V.dist = diag(as.matrix(V[c((length(theta) - 2 * K +
1):length(theta)), c((length(theta) - 2 * K + 1):length(theta))]))
for (i in seq(along = pi0)) {
        paramdist <- cbind(as.matrix(thetalinop[i, ]), as.matrix(sqrt
            ↪ (V.dist[c(1 *
        (i == 1) + 3 * (i == 2), 2 * (i == 1) + 4 * (i ==
        2))])))
        colnames(paramdist) <- c(paste0("estimate ", "(k=",
        i, ")"), paste0("s.e. ", "(k=", i, ")"))
        paramdistfinal <- cbind(paramdistfinal, paramdist)

}
```

```
# semi-Markov result for a 2-states
if (K > 2 & semi)
cat("Semi-Markovian approximation with more than 2 state is too
    ↪ demanding... please select K=2")
if (K == 2 & semi) {
        p1op <- c(log(1), logit(Pop[1, 2]))
        p2op <- c(log(1), logit(Pop[2, 1]))
        theta <- c(p1op, p2op, kappaop[1, ], kappaop[2, ],
        log(thetalinop[1, ]), log(thetalinop[2, ]))
        nb.target <- length(kappaop[1, ]) - 1
        m <- c(30, 30)
        w <- runif(sum(m))
        pi0 <- w/sum(w)
        minL = -exp(16)
        for (l in (1:nb_init)) {
                thetaop = theta + runif(length(theta), -1, 1)
                V3 <- optim(thetaop, fn = logLSemi, gr = NULL,
                y0 = y0, data = data, pi0 = pi0, nb.target = nb.
                    ↪ target,
                m = m, dist = dist, method = "L-BFGS-B", control =
                    ↪ list(fnscale = -1,
                maxit = 10^(8)), hessian = TRUE)
                if (V3$value > minL) {
                        minL = V3$value
                        V2 = V3
                }
        }
        V <- sqrt(diag(-(solve(V2$hessian)))) #actually solves the
            ↪ hessian to find the maxima and minima

        p1op <- c(exp(V2$par[1]), inv.logit(V2$par[2]))
```

```
sp1op <- c(p1op[1] * V[1], p1op[2] * (1 - p1op[2]) *
V[2])
p2op <- c(exp(V2$par[3]), inv.logit(V2$par[4]))
sp2op <- c(p2op[1] * V[3], p2op[2] * (1 - p2op[2]) *
V[4])

kappaop <- V2$par[c(5:((5 + 2 * nb.target + 1)))]
skappaop <- V[c(5:((5 + 2 * nb.target + 1)))]
n <- length(V)
thetalinop <- exp(V2$par[c((n - 3):n)])
sthetalinop <- thetalinop * V[c((n - 3):n)]
paramkappafinalS <- NULL
paramkappafinalS <- cbind(kappaop[1:(nb.target +
1)], skappaop[1:(nb.target + 1)], kappaop[(nb.target +
2):(2 * (nb.target + 1))], skappaop[(nb.target +
2):(2 * (nb.target + 1))])

colnames(paramkappafinalS) <- c("estimate (k=1)",
"s.e.", "estimate (k=2)", "s.e.")

Dwell <- rbind(c(p1op[1], sp1op[1], p1op[2], sp1op[2]),
c(p2op[1], sp2op[1], p2op[2], sp2op[2]))
colnames(Dwell) <- c("size", "s.e", "prob", "s.e")
rownames(Dwell) <- c("p1", "p2")

paramdistfinalS <- NULL
paramdistfinalS <- cbind(thetalinop[c(1, 2)], sthetalinop[c
    ↪ (1,
2)], thetalinop[c(3, 4)], sthetalinop[c(3, 4)])
colnames(paramdistfinalS) <- c("phi (k=1)", "s.e",
"phi (k=2)", "s.e")
```

```
        P <- gen.Gamma(m, cbind(c(p1op[1], p2op[1]), c(p1op[2],
        p2op[2])))
        Pa <- P
        kappaop <- rbind(t(paramkappafinalS[, 1]), t(paramkappafinalS
            ↪ [,
        3]))

        kappacopy <- NULL
        thetalincopy <- NULL
        for (i in (1:m[1])) {
                kappacopy <- rbind(kappacopy, kappaop[1, ])
                thetalincopy <- rbind(thetalincopy, thetalinop[1:2])
        }
        for (i in (1:m[2])) {
                kappacopy <- rbind(kappacopy, kappaop[2, ])
                thetalincopy <- rbind(thetalincopy, thetalinop[3:4])
        }
        param = list(pi0 = pi0, kappa = kappacopy, thetalin =
            ↪ thetalincopy,
        P = P)
        data <- list(x = x, y = y, z = z, d = d)
        FS <- FilterSmooth(param, data, type, dist)

        E.smooth0 <- FS$E.smooth0
        E.filter <- FS$E.filter
        E.smooth <- FS$E.smooth
        LL.prev <- FS$LogLike
}


fit.Markov = list(likelihood.Markov = LL, AIC.Markov = -2 *
LL + 2 * length(theta.Markov), BIC.Markov = -2 *
```

```
                LL + log(length(y)) * length(theta.Markov))
                if (semi)
                fit.SemiMarkov = list(likelihood.SemiMarkov = LL.prev,
                AIC.SemiMarkov = -2 * LL.prev + 2 * length(theta),
                BIC.SemiMarkov = -2 * LL.prev + log(length(y)) *
                length(theta))

                Markov <- list(param = param.Markov, P.Markov = paramPfinal,
                kappa.Markov = paramkappafinal, phi.Markov = paramdistfinal,
                fit.Markov = fit.Markov)
                if (semi)
                SemiMarkov <- list(Dwell.SemiMarkov = Dwell, kappa.SemiMarkov =
                    ↪ paramkappafinalS,
                phi.SemiMarkov = paramdistfinalS, fit.SemiMarkov = fit.SemiMarkov)
                out <- list(Markov = Markov, param = param, EM.itermax = s)
                if (semi)
                out <- list(Markov = Markov, param = param, SemiMarkov = SemiMarkov,
                EM.itermax = s)
        }
        class(out) <- "GHRandomWalk"
        out
}


#x An object, produced by the \code{\link{consensus}} function, to print.
# \dots Further arguments to be passed to \code{print.default}.
print.GlobalMaxima <- function(x, ...) {
        cat("\nMarkov specification:")
        cat("\nMax log-likelihood and classical criteria:", x$fit.Markov,
        "\n")
        cat("\nParameters of directions :\n")
        print.default(x$fit.Markov$kappa.Markov, print.gap = 2, quote = FALSE,
```

```
        right = TRUE, ...)
        cat("\n")
        cat("\nParameters of distances :\n")
        print.default(x$fit.Markov$phi.Markov, print.gap = 2,
        quote = FALSE, right = TRUE, ...)
        cat("\n")
        cat("\nTransition matrix :\n")
        print.default(x$fit.Markov$P.Markov, print.gap = 2, quote = FALSE,
        right = TRUE, ...)
        cat("\n")
        invisible(x)
}


######################## consensus 1 state
########################


#Consensus Model: Function that fit a consensus model for angular variables and
    ↪ its method.
#pginit: The approximate number of points on the grid of possible initial beta
    ↪ values tried when initbeta is not given. The default is 1000, which runs
    ↪ quickly. A large value of makes the function slower.
#maxiter: The maximum number of iterations. The default is 1000.
#mindiff: The minimum difference between two max cosine to be reached.
#formula: A formula with the dependent angle on the left of the ~ operator and
    ↪ terms specifying the explanatory variables on the right. These terms must
    ↪ be written x:z, where x is an explanatory angle which relative importance
    ↪ might depend on the positive variable z. It is not mandatory to specify a z
    ↪  variable for each explanatory angle. For model='simplified', the first
    ↪ explanatory angle listed is the reference direction (if a z variable was
    ↪ specified for this angle, it is ignored).
```

```
#MaxLL: the maximum value of the log likelihood
#parameters: the parameter estimates and their standard errors (obtained from two
    ↪ definitions)
#varcov1: the estimated variance covariance matrix for the parameter estimates (
    ↪ obtained from the first definition)
#varcov2: the estimated variance covariance matrix for the parameter estimates (
    ↪ obtained from the second definition)
#parambeta: the beta parameter estimates and their standard errors (obtained by
    ↪ linearization)
#varcovbeta1:the estimated variance covariance matrix for the beta parameter
    ↪ estimates (obtained by linearization)
#varcovbeta2:the estimated variance covariance matrix for the beta parameter
    ↪ estimates (Sandwhich form)
#autocorr: the autocorrelation of the residuals
#iter.detail:the iteration details
#converge: an indicator of convergence
#call: the function call

consensus <- function(formula, data, model = "simplified", weights = NULL,
    ↪ initbeta = NULL, control = list()) {

        call <- mfcall <- match.call()
        model <- model[1]

        ### information from formula and data
        mfargs <- match(c("formula", "data"), names(mfcall), 0L)
        mfcall <- mfcall[c(1L, mfargs)]
        mfcall[[1L]] <- as.name("model.frame")
        mf <- eval(mfcall, parent.frame())
        # useful objects
        nobs <- nrow(mf)
```

```
nomterms <- attr(attr(mf, "terms"), "term.labels")
nterms <- length(nomterms)
p <- if ("simplified" == model)
nterms - 1 else nterms
nparam <- if ("simplified" == model)
p + 1 else p + 2
# paramname <- paste0('kappa', 0:p)
paramname <- nomterms
if ("complete" == model)
paramname <- c(paramname, paste0("beta", p + 1))
# first column = response variable
y <- as.vector(mf[, 1])
# explanatory variables
noms <- strsplit(nomterms, split = ":")
noms <- do.call(rbind, noms)
if ("simplified" == model) {
       x0 <- mf[, noms[1, 1]]
       noms <- noms[-1, , drop = FALSE]
}
matx <- as.matrix(mf[, noms[, 1], drop = FALSE])
if (ncol(noms) == 1) {
       matz <- matrix(1, ncol = ncol(matx), nrow = nrow(matx)) # all z are
           ↪ 1
} else {
       matz <- as.matrix(mf[, noms[, 2], drop = FALSE])
       matz[, noms[, 2] == noms[, 1]] <- 1 # unspecified z are 1
}
weight = rep(1, nobs) * (is.null(weights)) + (!is.null(weights)) *
weights #this is the weights for the "weighted" von mises.

### log-likelihood function
LL <- function(param) {
```

```
        angleref <- if ("simplified" == model)
        x0 else rep(param[p + 2], nobs)
        # length of the vector
        #this is from V_t(k) eqtn 5
        sinmu <- param[1] * sin(angleref) + (matz * sin(matx)) %*%
        param[2:(p + 1)]
        cosmu <- param[1] * cos(angleref) + (matz * cos(matx)) %*%
        param[2:(p + 1)]
        long <- as.vector(sqrt(sinmu^2 + cosmu^2))
        # predicted value form the model
        mui <- as.vector(atan2(sinmu, cosmu))
        # log likelihood : of the consensus model
        term1 <- param[1] * cos(y - angleref) + (matz * cos(y -
        matx)) %*% param[2:(p + 1)]
        # LL <- sum(weight*term1) - sum(weight*log(besselI(long, 0,
        # expon.scaled = FALSE)))
        LL <- sum(term1) - sum(log(besselI(long, 0, expon.scaled = FALSE)))



        list(LL = LL, long = long, mui = mui)
}
# Function that update parameter of the log-likelihood
# function
paramUpdate <- function(paramk, long, mui) { #updating the parameters
        angleref <- if ("simplified" == model)
        x0 else rep(paramk[p + 2], nobs)
        matx0 <- cbind(angleref, matx)
        matz0 <- cbind(rep(1, nobs), matz)
        # score vector
        Along <- as.vector(besselI(long, 1, expon.scaled = FALSE)/besselI(
            ↪ long,
        0, expon.scaled = FALSE))
```

```
        matu <- matz0 * (cos(y - matx0) - cos(matx0 - mui) *
        Along)
        if ("complete" == model) {

            matu <- cbind(matu, paramk[1] * sin(y - angleref) -
            sin(mui - angleref) * Along) #along is a scaled modified
                ↪ bessel(ratio??)
        }
        vecs <- colSums(matu) #sum columnwise(downwards)
        names(vecs) <- paramname

        # Fisher information matrix
        Xc <- matz0 * cos(matx0 - mui)
        Xs <- matz0 * sin(matx0 - mui)
        if ("complete" == model) {

            Xc <- cbind(Xc, paramk[1] * sin(mui - paramk[p +
            2]))
            Xs <- cbind(Xs, paramk[1] * cos(mui - paramk[p +
            2]))
        }
        Dc <- diag(1 - Along/long - Along^2, nrow = nobs, ncol = nobs)
        Ds <- diag(Along/long, nrow = nobs, ncol = nobs)
        matI <- t(Xc) %*% Dc %*% Xc + t(Xs) %*% Ds %*% Xs
        colnames(matI) <- rownames(matI) <- paramname
        # update of parameters
        dparam <- as.vector(solve(matI, vecs))
        paramk1 <- paramk + dparam
        list(paramk1 = paramk1, dparam = dparam, matu = matu,
        matI = matI)
}
# initial values of parameters beta we try 10 000 differents
```

```
# possible values

#this is random selection of initial values with 10000 different options
if (is.null(initbeta)) {

       #taking both the complete and simplified model options into account

       pginit <- if (is.null(control$pginit))
       1000 else control$pginit
       pg <- round(pginit^(1/nparam))
       possparam <- rep(list(seq(-1, 1, length.out = pg + 2)[-c(1,
       pg + 2)]), p + 1)
       if ("complete" == model)
       possparam[[nparam]] <- seq(0, 2 * pi, length.out = pg +
       2)[-c(1, pg + 2)]
       possVal <- cbind(expand.grid(possparam), NA)
       colnames(possVal) <- c(paramname, "LL")
       maxLL <- function(param) LL(param = param)$LL
       possVal[, nparam + 1] <- apply(possVal[, 1:nparam], 1,
       maxLL)
       paramk <- unlist(possVal[which.max(possVal[, nparam +
       1]), 1:nparam])
} else {
       if (length(initbeta) != nparam)
       stop("for the requested model, 'initparam' must be of length ",
       nparam)
       paramk <- initbeta
}
# log likelihood value with respect to initial parameter
calcul <- LL(param = paramk)
maxLLk <- calcul$LL
long <- calcul$long
```

```
mui <- calcul$mui
# Initialization
iter <- iter.sh <- 0
maxiter <- if (is.null(control$maxiter))
1000 else control$maxiter
mindiff <- if (is.null(control$mindiff))
1e-06 else control$mindiff
conv <- FALSE
# Initialisation of the matrix of informations during the
# iterations
iter.detail <- matrix(NA, nrow = maxiter + 1, ncol = nparam +
3)
colnames(iter.detail) <- c(paramname, "maxLL", "iter", "nitersh")
iter.detail[1, ] <- c(paramk, maxLLk, iter, iter.sh)
# start of the fit
while (!conv && iter <= maxiter) {
       # update of the parameters
       maj <- paramUpdate(paramk = paramk, long = long, mui = mui)
       paramk1 <- maj$paramk1
       dparam <- maj$dparam
       # computation of the log-likelihood
       calcul <- LL(param = paramk1)
       maxLLk1 <- calcul$LL
       long <- calcul$long
       mui <- calcul$mui
       # if the criteria as decrease then do step halving
       iter.sh <- 0
       while (maxLLk1 < maxLLk) {
              iter.sh <- iter.sh + 1
              paramk1 <- paramk + dparam/(2^iter.sh)
              calcul <- LL(param = paramk1)
              maxLLk1 <- calcul$LL
```

```
                long <- calcul$long
                mui <- calcul$mui
                if (iter.sh >= maxiter)
                break
        }
        #Accounting for possible non-convergence
        # Does the criteria increase more than mindiff?
        if (maxLLk1 < maxLLk) {
                conv <- FALSE
                warning("the algorithm did not converge, it failed to
                    ↪ maximize the log likelihood")
                break
        } else {
                conv <- if (maxLLk1 - maxLLk > mindiff)
                FALSE else TRUE
                paramk <- paramk1
                maxLLk <- maxLLk1
                iter <- iter + 1
                iter.detail[iter + 1, ] <- c(paramk, maxLLk, iter,
                iter.sh)
        }
}
if (iter > maxiter + 1) { #we do not want to run more than the stipulated
    ↪ amount of simulations
        warning("the algorithm did not converge, the maximum number of
            ↪ iterations was reached")
} else {
        iter.detail <- iter.detail[1:(iter + 1), , drop = FALSE]
}

### Computation of standard errors and Fisher information
### matrix for the final parameters.
```

```
if (maxLLk == maxLLk1) {
        maj <- paramUpdate(paramk = paramk, long = long, mui = mui)
}

matu <- maj$matu
matI <- maj$matI
# parametric estimation of the covariance matrix
v1 <- solve(matI) #we obtain the covariance matrix here
# non parametric estimation
mid <- matrix(0, ncol = nparam, nrow = nparam)
for (i in 1:nobs) {
        mid <- mid + t(matu[i, , drop = FALSE]) %*% matu[i, ,
        drop = FALSE]
}
v2 <- v1 %*% mid %*% v1

### Results for the betas
paramb <- paramk[2:(p + 1)]/paramk[1]
matDeriv <- rbind(-paramk[2:(p + 1)]/paramk[1]^2, diag(1/paramk[1],
nrow = p, ncol = p))
vb <- t(matDeriv) %*% v1[1:(p + 1), 1:(p + 1)] %*% matDeriv
vb2 <- t(matDeriv) %*% v2[1:(p + 1), 1:(p + 1)] %*% matDeriv
# names(paramb) <- colnames(vb) <- rownames(vb)
# <-colnames(vb2) <- rownames(vb2)<- paste0('beta', 1:p)
names(paramb) <- colnames(vb) <- rownames(vb) <- colnames(vb2) <- rownames(
    ↪ vb2) <- paramname[-1]

### Output
zvalue <- abs(paramk)/sqrt(diag(v1))
p <- round(2 * pnorm(abs(paramk)/sqrt(diag(v1)), lower.tail = FALSE),
5)
parameters <- cbind(paramk, sqrt(diag(v1)), zvalue, p)
```

```
        # colnames(parameters) <- c('estimate', paste('stderr', 1:2,
        # sep=''))
        colnames(parameters) <- c("estimate", "stderr", "z value",
        "P(|z|>.)")
        rownames(parameters) <- paramname
        parambeta <- cbind(paramb, sqrt(diag(vb)), sqrt(diag(vb2)))
        colnames(parambeta) <- c("estimate", paste("stderr", 1:2,
        sep = ""))
        res <- sin(y - mui)
        autocorr <- acf(res, plot = FALSE)
        out <- list(MaxLL = maxLLk, parameters = parameters, varcov1 = v1,
        varcov2 = v2, parambeta = parambeta, varcovbeta1 = vb,
        varcovbeta2 = vb2, autocorr = autocorr, matx = matx,
        matz = matz, y = y, long = long, mui = mui, iter.detail = iter.detail,
        call = call)
        class(out) <- "consensus"
        out
}


#' @param x An object, produced by the \code{\link{consensus}} function, to print.
#' @param \dots Further arguments to be passed to \code{print.default}.
print.consensus <- function(x, ...) {
        cat("\nMaximum log-likelihood :", x$MaxLL, "\n")
        cat("\nParameters:\n")
        print.default(x$parameters, print.gap = 2, quote = FALSE,
        right = TRUE, ...)
        cat("\n")
        # cat('\nBeta Parameters:\n') print.default(x$parambeta,
        # print.gap = 2, quote = FALSE, right=TRUE, ...) cat('\n')
        invisible(x)
}
```

```
##### plot smooth proba on the trajectory (K=2 states)
#####



# Plot smooth probabilities on trajectory
#Function that smooth eacgh step of the trajectory by the smooth probabilites:
    ↪ This plot can be used to characterize spatially the behaviour of the animal
    ↪ .
#param: list of all the parameters of the model in this order: P, kappa and
    ↪ thetalin.
#data: data frame of the directions (y), distances (d), explanatory angles
    ↪ variables (x) and explanatory real variables (z)
#long: vectors of longtitude coordinates of the trajectory (x axis)
# lat: vectors of latitude coordinates of the trajectory (y axis)

PlotEstim <- function(data, param, long, lat, type = "angle-dist",
dist = "gumbel") {



       K <- length(param$pi0)
       p <- length(param$kappa[1, ]) - 1
       steps <- length(data$y)
       FS <- FilterSmooth(param, data, type, dist) #get the smoothed probabilities
       E.smooth <- FS$E.smooth
       explo <- 1
       position <- cbind(long, lat)
```

```
        Sx1 <- max(position[, 1]) - min(position[, 1])
        Sy1 <- max(position[, 2]) - min(position[, 2])
        x1 <- seq(min(position[, 1]) - 0.1 * Sx1, max(position[,
        1]) + 0.1 * Sx1, length.out = 1024)
        y1 <- seq(min(position[, 2]) - 0.1 * Sy1, max(position[,
        2]) + 0.1 * Sy1, length.out = 1024)
        Map <- matrix(1, length(x1), length(y1))
        image(x1, y1, (Map), col = "white")

        n <- 5
        colorter <- rgb(c(seq(0, 1, 1/n), rep(1, n)), c(abs(seq(0,
        1 - 1/n, 1/n)), 1, abs(seq(1 - 1/n, 0, -1/n))), c(rep(1,
        n), abs(seq(1, 0, -1/n))))

        for (i in (2:(length(position[, 1]) - 1))) {
                classe <- floor(E.smooth[i, explo] * 10) + 1
                segments(position[i - 1, 1], position[i - 1, 2], position[i,
                1], position[i, 2], lwd = 1, col = colorter[classe])
        }
}
```

```
library(boot)
library(circular)
library(Renext)
library(pracma)
library(fitdistrplus)
library(VGAM)
library(LindleyR)
library(Renext)
library(maxLik)
library(tictoc)
# Please load the appropriate workspace
```

```
tic("Model Run Time:")
source("Core Paper Main 2.R")
## simply importing the data into R
jeu <- data.frame(read.table(file = "caribou.txt", header = TRUE))
# initialization parameter of the EM algorithm with a one state model
n <- length(jeu$y);
#this is the precision of the movement. The -n is a lag n times before the current
    ↪ index
jeu$yprec <- c(0, jeu$y[ - n]); # we now have a column with the data in y starting
    ↪ from zero and the rest lagged from one spot

mc <- consensus(formula = y ~ yprec + xcut + xcenter, data = jeu);

#parameters column 1: estimate
#parameters column 3: z-value
kappac <- cbind(mc$parameters[, 1], mc$parameters[, 3]);

dmom <- ini.mixexp2(jeu$d); #this will give us an estimate(mixing probability) and
    ↪ two rates

mel.d <- mledist(jeu$d, distr = "gamma"); #this function fits a univariate
    ↪ distribution using mle for gamma distribution

#Gamma
gamma_func<- function(aa){
         shape1<- aa[1]
         scale1<- aa[2]
         sum(dgamma(jeu$d,shape = shape1,scale = scale1,log=TRUE))
       }
m<- maxLik(gamma_func, start = c(1,1),method = "NR")
summary(m)
```

```
#Weibull
wei_func<- function(aa){
        shape1<- aa[1]
         scale1<- aa[2]
         sum(dweibull(jeu$d,shape = shape1,scale = scale1,log=TRUE))
       }
m<- maxLik(wei_func, start = c(1,1),method = "NR")
summary(m)

# Power lindley
pl_func<- function(aa){
       shape1<- aa[1]
       scale1<- aa[2]
       sum(dplindley(jeu$d,theta = shape1,alpha = scale1,log=TRUE))
}
m<- maxLik(pl_func, start = c(1,1),method = "NR")
summary(m)

# Gumbel Distribution
gumbel_func<- function(aa){
       loc <- aa[1]
       scale1<- aa[2]
       sum(dgumbel(jeu$d, loc, scale1, log = TRUE))
}
m<- maxLik(gumbel_func, start = c(1,1),method = "NR")
summary(m)

pi0 <- c(0.7, 0.3); #initial distribution parameters

#these are the starting points for kappa1 and kappa2, from the initial estimation
    ↪ above.
```

```
kappainit <- rbind(c(kappac[1, 1], kappac[2:3, 1]),
c(kappac[1, 1], kappac[2:3, 1]));

#these are the random starting points for the phi's from the initial estimate
    ↪ above.
#this will be the gamma parameters used for further estimation

#This is the one you use when you have Weibull and Gamma!!
#thetalininit <- rbind(c(mel.d$estimate[1], 1/mel.d$estimate[2]),
# c(mel.d$estimate[1], 1/mel.d$estimate[2]))

# This is the one you use when you have the lindley and Gumbel
thetalininit <- rbind(c(m$estimate[1], m$estimate[2]),
c(m$estimate[1], m$estimate[2]))

#colnames(thetalininit) <- c("shape", "scale");

#for when we have Power Lindley
colnames(thetalininit) <- c("scale", "shape");

paraminitiaux <- list(pi0 = pi0, kappainit = kappainit,
thetalininit = thetalininit, Pinit = NULL);

# data set for the fit

target <- cbind(jeu$xcut, jeu$xcenter);
p <- length(target[1, ]); #number of states we will be dealing with
z <- matrix(1,n,p); #matrix of 1's, with n rows and p columns i,e, 617x2 matrix of
    ↪  1's
jeu <- list(x = as.matrix(target), y=as.vector(jeu$y),
z = z, d = as.vector(jeu$d))
```

```
# FIT
K <- 2 #states
#The GHRandomWalk will fit the actual model
GM <- GHRandomWalk(data = jeu, K, paraminitiaux, nb_init = 5, dist = "gumbel");
GM
toc()

### My tabular things to reproduce the tabular output
table_results<- rbind(
round(c(1-GM$Markov$P.Markov[1],GM$Markov$P.Markov[2]),4),
c(1,"-"),
round(c(GM$Markov$P.Markov[3],GM$Markov$P.Markov[3]),4),
c(1,"-"),

round(c(GM$Markov$kappa.Markov[1],GM$Markov$kappa.Markov[4]),4),
round(c(GM$Markov$kappa.Markov[2],GM$Markov$kappa.Markov[5]),4),
round(c(GM$Markov$kappa.Markov[3],GM$Markov$kappa.Markov[6]),4),

round(c(GM$Markov$phi.Markov[1],GM$Markov$phi.Markov[3]),4),
round(c(GM$Markov$phi.Markov[2],GM$Markov$phi.Markov[4]),4),

round(c(GM$Markov$kappa.Markov[10],GM$Markov$kappa.Markov[13]),4),
round(c(GM$Markov$kappa.Markov[11],GM$Markov$kappa.Markov[14]),4),
round(c(GM$Markov$kappa.Markov[12],GM$Markov$kappa.Markov[15]),4),

round(c(GM$Markov$phi.Markov[5],GM$Markov$phi.Markov[7]),4),
round(c(GM$Markov$phi.Markov[6],GM$Markov$phi.Markov[8]),4),

c(round(GM$Markov$fit.Markov$likelihood.Markov,4),"-"),
c(round(GM$Markov$fit.Markov$AIC.Markov,4),"-"),
c(round(GM$Markov$fit.Markov$BIC.Markov,4),"-"))
```

```
table_results<- as.data.frame(table_results)


steps <- length(jeu$y); #length of data i.e. 617 steps
position <- matrix(0, steps, 2); #matrix of 617x2 of zeros
position[1, ] <- c(0, 0); #starting position is (0,0): x y co-ordinates
for (i in (2:steps)) #we exclude the first position since it is zero
{
       position[i, ] = position[i - 1, ] + jeu$d[i] * 1000 * cbind(cos(jeu$y[i]),
           ↪ sin(jeu$y[i]));
}
PlotEstim(jeu, param = GM$param, long = position[, 1], lat = position[, 2])
box()
```

References

[1] Toshihiro Abe and Arthur Pewsey. Sine-skewed circular distributions. *Statistical Papers*, 52(3):683–707, 2011.

[2] Mohammad Ahsanullah and Mohammed Zafar Anis. Characterization of sine- skewed von mises distribution. *arXiv: Other Statistics*, 2019.

[3] Lee J. Bain and Max Engelhardt. *Introduction to probability and mathematical statistics*. Brooks/Cole, 1987.

[4] Christophe Biernacki, Gilles Celeux, and Gerard Govaert. Choosing starting values for the EM algorithm for getting the highest likehood in multivariate Gaussian mixture models. *Computational Statistics and Data Analysis*, 41:561–575, 01 2003. doi: 10.1016/S0167-9473(02)00163-9.

[5] Jeff Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, 1998.

[6] Moritz Blume. Expectation Maximization: A gentle introduction. *Technical University of Munich Institute for Computer Science*, 2002.

[7] Carlos Bruno Briozzo, Carlos E. Budde, Omar Osenda, and Manuel O. Cáceres. Exact and asymptotic properties of multistate random walks. *Journal of Statistical Physics*, 65:167–182, 1991.

[8] Subrata Chakraborty and Dhrubajyoti Chakravarty. A discrete Gumbel distribution. *arXiv:1410.7568 [math.ST]*, 2014.

[9] Edward A. Codling, Michael J. Plank, and Simon Benhamou. Random walk models in biology. *Journal of The Royal Society Interface*, 5(25):813–834, 2008. doi: 10.1098/rsif.2008.0014. URL `https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2008.0014`.

[10] Jolien Cremers and Irene Klugkist. One Direction? A Tutorial for Circular Data Analysis Using R With Examples in Cognitive Psychology. *Frontiers in Psychology*, 9, 2018. ISSN 1664-1078. doi: 10.3389/fpsyg.2018.02040. URL `https://www.frontiersin.org/article/10.3389/fpsyg.2018.02040`.

[11] Thierry Duchesne, Daniel Fortin, and Louis-Paul Rivest. Equivalence between step selection functions and biased correlated random walks for statistical inference on animal movement. *PLOS ONE*, 10(4):1–12, 04 2015. doi: 10.1371/journal.pone.0122947. URL `https://doi.org/10.1371/journal.pone.0122947`.

[12] Daniel Fortin, Pietro-Luciano Buono, André Fortin, Nicolas Courbin, Christian Tye Gingras, Paul R. Moorcroft, Réhaume Courtois, and Claude Dussault. Movement responses of caribou to human-induced habitat edges lead to their aggregation near anthropogenic features. *The American Naturalist*, 181(6):827–836, 2013.

[13] Sylvia Frühwirth-Schnatter. *Finite mixture and Markov switching models*. Springer Science and Business Media, 2006.

[14] M. Ghitany, D. Al-Mutairi, N. Balakrishnan, and L. Al-Enezi. Power Lindley distribution and associated inference. *Computational Statistics and Data Analysis*, 64:20–33, 2013.

[15] V. Goodall, L. Fatti, and N. Owen-Smith. Animal movement modelling: independent or dependent models? *South African Statistical Journal*, 51:295–315, 12, 2017.

[16] A. Lawrence Gould. A regression technique for angular variates. *Biometrics*, pages 683–700, 1969.

[17] Izrail Solomonovich Gradshteyn and Iosif Moiseevich Ryzhik. *Table of integrals, series, and products*. Academic press, 2014.

[18] Tim Hesterberg. Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(6):497–526, 2011.

[19] Oliver Ibe. *Markov processes for stochastic modeling*. Newnes, 2013.

[20] Richard A. Johnson and Thomas E. Wehrly. Some angular-linear distributions and related regression models. *Journal of the American Statistical Association*, 73(363):602–606, 1978.

[21] Morteza Khodabina and Alireza Ahmadabadib. Some properties of generalized gamma distribution. 2010.

[22] Samuel Kotz and Saralees Nadarajah. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[23] Jeffrey Lagarias. Euler's constant: Euler's work and modern developments. *Bulletin of the American Mathematical Society*, 50(4):527–628, 2013.

[24] Roland Langrock, Ruth King, Jason Matthiopoulos, Len Thomas, Daniel Fortin, and Juan M. Morales. Flexible and practical modeling of animal telemetry data: Hidden Markov models and extensions. *Ecology*, 93(11):2336–2342, 2012. doi: 10.1890/11-2241.1. URL `https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/11-2241.1`.

[25] Guillaume Latombe, Lael Parrott, Mathieu Basille, and Daniel Fortin. Uniting statistical and individual-based approaches for animal movement modelling. *PLOS ONE*, 9(6):1–12, 06 2014. doi: 10.1371/journal.pone.0099938. URL `https://doi.org/10.1371/journal.pone.0099938`.

[26] Gregory F. Lawler and Vlada Limic. *Random walk: a modern introduction*. Cambridge University Press, 2010.

[27] Smail Mahdi and Myrtene Cenac. Estimating parameters of gumbel distribution using the methods of moments, probability weighted moments and maximum likelihood. *Revista de Matemática: Teoría y Aplicaciones*, 12:151, 02 2012. doi: 10.15517/rmta.v12i1-2.259.

[28] Kanti V. Mardia and Peter E. Jupp. *Directional statistics*. John Wiley and Sons, 2009.

[29] Guillaume Moreau, Daniel Fortin, Serge Couturier, and Thierry Duchesne. Multi-level functional responses for wildlife conservation: the case of threatened caribou in managed boreal forests. *Journal of Applied Ecology*, 49(3):611–620, 2012. doi: 10.1111/j.1365-2664.2012.02134.x. URL `https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2664.2012.02134.x`.

[30] Loc Nguyen. Tutorial on EM Algorithm. 10 2020. doi: 10.20944/preprints201802.0131.v8.

[31] Aurélien Nicosia, Thierry Duchesne, Louis-Paul Rivest, and Daniel Fortin. 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(3):445–463, 2016.

[32] Aurélien Nicosia, Thierry Duchesne, Louis-Paul Rivest, and Daniel Fortin. A general hidden state random walk model for animal movement. *Computational Statistics and Data Analysis*, 105:76–95, 2017. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2016.07.009. URL `http://www.sciencedirect.com/science/article/pii/S0167947316301736`.

[33] Aurélien Nicosia, Thierry Duchesne, Louis-Paul Rivest, Daniel Fortin, et al. A multi-state conditional logistic regression model for the analysis of animal movement. *The Annals of Applied Statistics*, 11(3):1537–1560, 2017.

[34] Fernanda V. Paula, Abraao D. C. Nascimento, and Getúlio J. A. Amaral. A new extended Cardioid model: an application to wind data. *arXiv preprint arXiv:1712.01824*, 2017.

[35] Mark A. Pinsky and Samuel Karlin. Markov Chains: Introduction. In Mark A. Pinsky and Samuel Karlin, editors, *An Introduction to Stochastic Modeling (Fourth Edition)*, pages 79–163. Academic Press, Boston, fourth edition edition, 2011. ISBN 978-0-12-381416-6. doi: https://doi.org/10.1016/B978-0-12-381416-6.00003-4. URL `http://www.sciencedirect.com/science/article/pii/B9780123814166000034`.

[36] Brett Presnell, Scott P. Morrison, and Ramon C. Littell. Projected multivariate linear models for directional data. *Journal of the American Statistical Association*, 93(443):1068–1077, 1998.

[37] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[38] Louis-Paul Rivest. A decentred predictor for ciruclar-circular regression. *Biometrika*, 84(3):717–726, 1997. ISSN 00063444. URL http://www.jstor.org/stable/2337591.

[39] Ali Esmaieeli Sikaroudi and Chiwoo Park. A mixture of linear-linear regression models for a linear-circular regression. *Statistical Modelling*, (0), 2016. doi: 10.1177/1471082X19881840. URL https://doi.org/10.1177/1471082X19881840.

[40] Bracken Van Niekerk. Application of hidden markov models and their extensions to animal movement data. Master's thesis, Nelson Mandela University, 2018.

[41] Shun-Zheng Yu. *Hidden Semi-Markov models: theory, algorithms and applications*. Morgan Kaufmann, 2015.