A Spatio-Temporal Model for Arctic Sea Ice

Alison Kleffner 1
Department of Statistics, University of Nebraska - Lincoln
and
Susan VanderPlas 2
Department of Statistics, University of Nebraska - Lincoln
and
Yawen Guan 3
Department of Statistics, University of Nebraska - Lincoln

September 9, 2022

Abstract

Arctic Sea Ice is a barrier between the warm air of the ocean and the atmosphere, thus playing an important role in the climate. When narrow linear cracks (leads) form in the sea ice, the heat from the ocean is then released into the atmosphere. To estimate where cracks may form, motion data from the RADARSAT Geophysical Processing System (RGPS) are analyzed. The RGPS provides a set of trajectories (or cells) to trace the displacements of sea ice, however, chunks of data are missing due to the data collection method. We propose a spatial clustering and interpolation method that allows us to infer missing observations and estimate, where a crack may form. To do this feature inputs were created for KNN clustering by creating a bounding box around each trajectory, resulting in trajectories being assigned a cluster. A crack is considered to have formed on the boundary between different clusters. Within the clusters, spatiotemporal interpolation method is used to infer missing locations. Our clustering approach is then compared to other methods to determine ice crack formation, and cross-validation is used to assess our interpolation method.

Keywords: spatial clustering, non-stationary, Gaussian process.

1 Introduction

Sea ice is frozen sea water in the Arctic Ocean that generally occurs as an ice pack which can drift over the oceans surface. Understanding ice dynamics plays an vital role in climate models as it acts as a barrier between the colder atmosphere and the warmer ocean. Therefore, sea ice plays a role in the Earth's energy balance. Cracks in the ice can form due to dynamic processes, which includes wind, ocean currents, and tides (Hutter et al. 2019). When a narrow linear crack, or lead, forms in the sea ice, it allows for heat from the ocean to be transferred to the atmosphere (Schreyer et al. 2006). Only small amounts of the ice cover in the winter are occupied by leads (1-2%), but leads account for half of the heat flux between the ocean and the atmosphere (Badgley 1961). Additionally, leads are understood to be sources of global methane emissions, which makes them a possible force for greenhouse gasses (Kort et al. 2012). Hence, it is important to be able to determine where these leads may form in order to accurately account for their impact in climate models.

Previous methods to determine lead locations involve the use of deformation calculations and thermal information from satellite images. Drawbacks of satellite images include being low in resolution and are affected by atmospheric conditions, like clouds (Key et al. 1993). Thus, results may be impacted by errors due to inaccurate data. For example, errors in deformation calculations may lead to incorrect locations of ice cracks (Bouillon & Rampal 2015). Our approach to discover cracks uses the idea that sea ice is not stationary, as ocean currents and atmospheric winds drive its movement (Peterson & Sulsky 2011). Hence, using only data on the movement of an ice sheet, we hope to determine the location of sea ice cracks through the clustering of similar movements in the sea ice. We hypothesize that cracks will form at the location of the boundaries between clusters, as this is the boundary between different movements in the ice sheet.

An additional complication is that satellite data is often missing in chunks because the satellite may not pass over a certain region on a day. Thus, in order to have complete gridded data, we also developed an interpolation method that uses information given by our clusters to estimate missing regions.

1.1 Data

Motion data of the ice sheet comes from the RADARSTAT Geophysical Processing System (RGPS). In general, this data is collected through the use of RADARSTAT synthetic aperture radar (SAR) images obtained by satellites. The SAR images are then processed and produce estimates of items, like sea ice motion (Lindsay & Stern 2003). In order to track movement, an initial grid is created at the start of the study period, where each cell dimension is 10 km on a side (Kwok & Cunningham 2002). The vertices of each cell are assigned an identifier. These points will be referred to as gpids (g) moving forward. The trajectory of the sea ice is then found by tracking these gpids in sequential radar images (Kwok & Cunningham 2002). Typically the tracking continues until the melting season begins. The set of trajectories of all the gpids may be formally represented as $\mathcal{T} = (g_1, ..., g_n)$, where $g_k = <s_{1,j}, ..., s_{t,j}>$ and $s_{i,j} = (x_{i,j}, y_{i,j})$. In other words, in a set of n gpid trajectories each g_k is the trajectory of a gpid, k, for days i = 1,. . . ,t, with t being the last day, and k = 1,. . . ,n. Additionally, if considering the time period as a day, a gpid may have more than one observation for a day as the satellite may have passed over the region at multiple times. Thus, our notation includes j to account for a day potentially having multiple observations. We developed our methods using 22 days worth of data (t=22) where a difference of one integer means those observations are one day apart. We are using a small data set in order to see if methodology can be developed where not a lot of data is used in order for computational feasibility.

1.2 Challenges

Determining the locations of sea ice cracks is challenging with the given data for multiple reasons. First, we want to develop our methods only using motion data of the ice sheet. That is, using only the location of gpids at different time points. Secondly, due to obtaining the data through sequential satellite images, data may be missing because the satellite may not have passed over part of the sea ice at a given time resulting in chunks of missing data. Fig. 1 shows a plot of the ice sheet on the first day in the data. Each box is a gpid and the unfilled boxes (imputed = TRUE) are gpids that are unobserved at that time. The location of the unfilled boxes in the plot is its closest known location. In this image, there

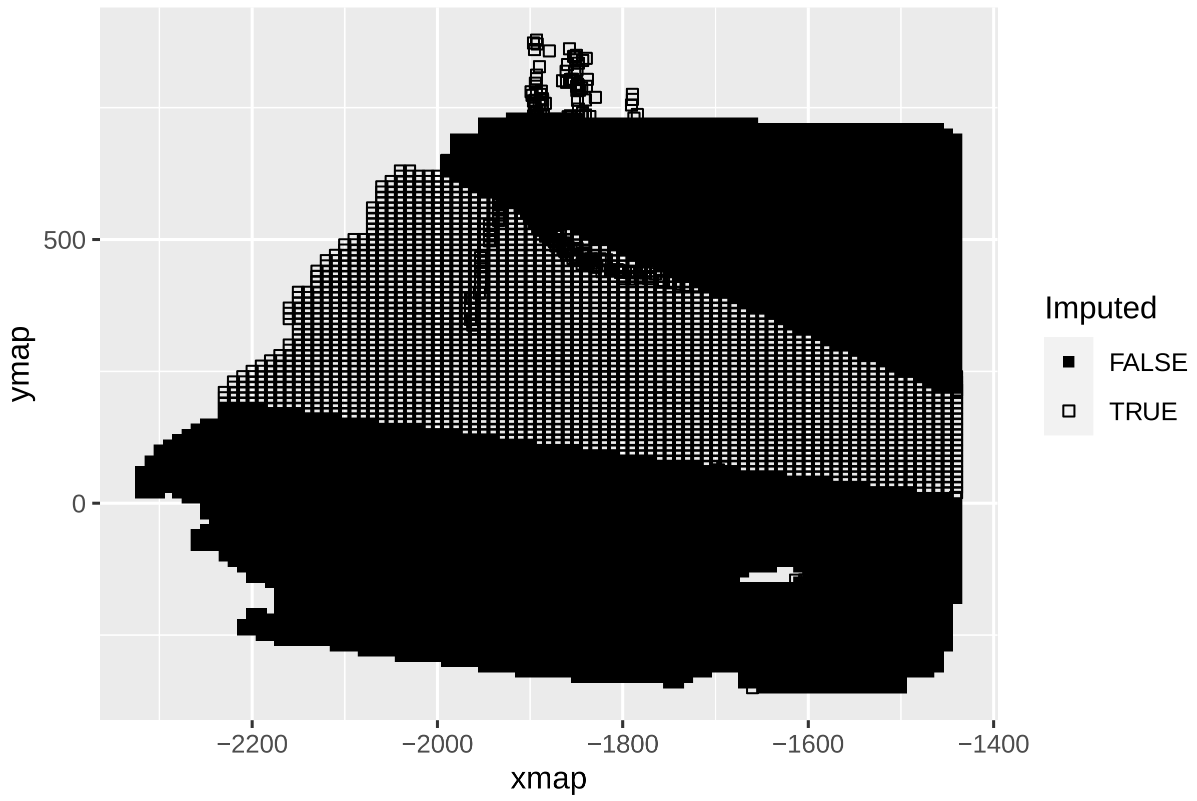

Figure 1: Plot of gpids at t=1. Missing data is represented by unfilled boxes, and is show to be missing in chunks instead of at random points

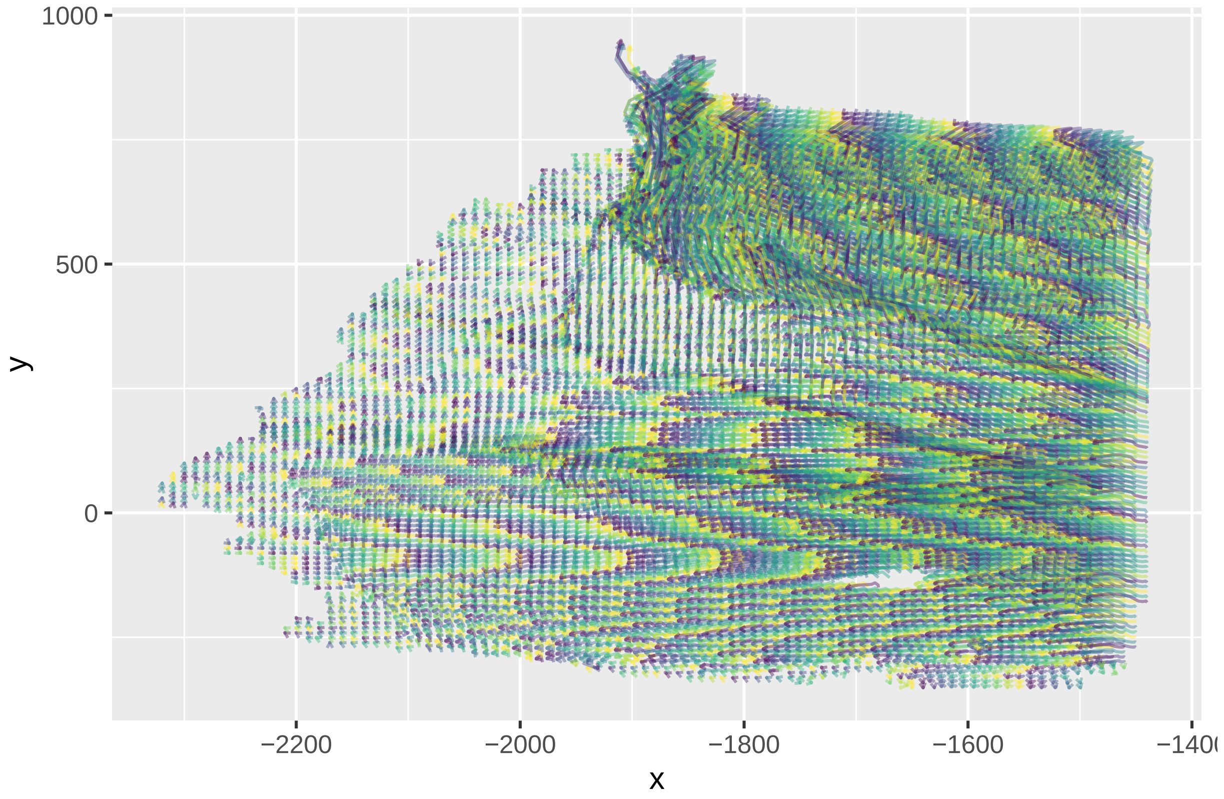

Figure 2: Plot of gpid trajectories over time to show movement and direction of movement of each gpid

is a large patch of data missing in the middle of the region, which makes it difficult to know what is happening in those areas as there may not be a neighbor we can infer from. Additionally, the missing data creates gpid trajectories of different lengths, which are not allowed in many standard trajectory distance functions. The third challenge is that the gpids were created on a grid, disqualifying the use of many popular spatio-temporal data mining methods based on the density of observations as point density will be consistent across the domain.

1.3 Nonstationary Spatio-Temporal Methods

1.3.1 Method Motivation

Since we want to detect cracks only using movement data, it is important to first see what the movement looks like. Thus, in order to visualize the movement of each gpid over the time frame, a plot of the trajectory of each gpid was created and is shown in Fig. 2. Each observed gpid location is connected with an arrow to indicate movement direction. The colors are used to distinguish individual trajectories but, otherwise they do not have a special meaning. This figure shows that groups of gpids appear to move with similar patterns. For instance, at the top of Fig. 2, a group of gpids seem to be traveling upwards, while in the middle a group is traveling from right to left. Visualizing these movements led to the idea of clustering similar trajectories. Trajectories can be summarized by a bounding box that represents its movement over time. This allows for the creation of features that can be used in clustering algorithms and it circumnavigates the missing data issue since trajectories do not need to be the same length. After the boundary box features are created, the gpid trajectories can then be grouped together with others that have similar features using a clustering algorithm. The idea is that sea ice cracks may form on the border between clusters, as gpids in different clusters have different movements, which may cause them to move apart.

2 Methods

2.1 Spatio-Temporal Clustering: Bounding Box

2.1.1 Bounding Box Features

The bounding box created around each gpid trajectory, which represent its movement, is made up of a number of different features. Bounding boxes can be created around a whole trajectory, or for smaller sub-trajectories, like a week. Sub-trajectories may provide more information while ensuring adequate data is available for the clustering process. A balance between ensuring data completeness and capturing sufficient movement is essential. An example of the points used to calculate the features of a trajectory may be found in **??**. Features of the bounding box includes the length of x traveled and the length of y traveled between the maximum and minimum location ($x_{max} - x_{min}$ and $y_{max} - y_{min}$), representing the total distance traveled. However, the maximum and minimum locations may not always correspond to the first and last days of the time frame. Hence, the difference in x and y from the latest observation to the earliest observation in a time period ($x_1 - x_0$ and $y_1 - y_0$) was also found. The change from latest to earliest observation is then used to calculate the angle of change in order to find the direction of movement of the trajectory. Further, since the focus is on the cluster boundaries, we must ensure the clusters are continguous, meaning clusters are not co-located across multiple geographic locations. To ensure some geographic continuity, we also include the average x and y value for each gpid. Finally, creating clusters for subtrajectories, for instance by week, can also be accomplished using a bounding box. However, previous week features can also be included as inputs into the clustering algorithm. This may be done in order for there to be some consistency between consecutive time frames, as previous movement may impact current movement. After all of the different features were calculated, the values were then standardized to give each feature similar weight in the clustering process.

2.1.2 K-Means

We used k-means clustering, which partitions n observations into k clusters with k being a pre-specified number. The goal of this clustering method is to minimize the squared Euclidean distance between an observation and the centroid vector of a cluster. The centroid vectors are found by averaging the features of each cluster member. K-means clustering is an iterative procedure:

1. The k clusters are given initial vectors.
2. The Euclidean distance between the i^{th} observation and k^{th} initial vector is obtained, and the observations are assigned to the cluster in which they have the smallest distance.
3. After all observations are allocated, the centroid vector for each cluster is then calculated by averaging each feature.
4. Observations are moved to a different cluster when appropriate (ie. has minimum Euclidean distance to a different cluster's centroid vector).
5. The centroid vector of the clusters is recalculated.

These steps are repeated until observations are no longer moving between clusters (Steinley 2006).

A drawback of k-means is that the number of clusters must be known and specified prior to clustering. We determine the number of clusters using the Silhouette statistic, which compares within cluster distances to between cluster differences. The silhouette width is defined as

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \quad (1)$$

where i is an observation and i $\in$ n. The value a(i) is the average distance between i and the other observations in the same cluster, and b(i) is the minimum average distance between i and the observations in other clusters. Silhouette width ranges from -1 to 1, with -1 meaning it was misclustered and 1 meaning it is well-clustered. The desired number of clusters is then determined by the largest average silhouette width (Kodinariya & Makwana 2013). Ice movement is a dynamic process, so if clustering subtrajectories, like weeks, then would expect that each week may have a different number of clusters. Thus, when clustering by week, the silhouette statistic will be calculated separately for each week.

2.2 Spatio-Temporal Interpolation

2.2.1 Finding Spatio-Temporal Neighbors

Next, as seen previously, due to the data collection method, the data is susceptible to be missing in chunks. Hence, want to be able to interpolate these missing data points in order to have completely gridded data. This can be challenging due to the lack of close neighbors, as all the data around a point may also be missing. Additionally, some interpolation methods are not available due to the nonstationarity of the data, as the ice is moving in patches, with the patches determined by our clustering method.

Our proposed method for interpolating missing gpid locations involves finding and using the spatial-temporal neighbors of the missing gpid. The clusters determined by our bounding box method can be used to identify these spatio-temporal neighbors. Creating the clusters is determined by similar movements, so can use the information given to us by these clusters in order to interpolate. Meaning, if we know how points move in a cluster at a specific time, we can assume a missing gpid in that same cluster would move similarly. In this method, new clusters are created for each week. The idea is that the intersection of one week's clusters with the week before and week after would create groups. Each member of a group is then a spatio-temporal neighbor of the other members as they are in a similar geographic region over time.

The first step in order to find these intersections is to create polygons for each of the clusters for each week. A polygon is created by finding the boundary coordinates of each of the clusters. In a sequential manner, a polygon is created around each cluster, generally starting with one on the top edge of the ice sheet followed by neighboring clusters until there is a polygon for each cluster. After each polygon is created, then all of the gpids that are located within this polygon are then removed from the data set, even if the gpid was assigned to a different cluster. This was done to reduce the amount of polygon overlapping. Additionally, if a gpid is classified to a different cluster than all of its neighbors, most likely that gpid should actually be classified like its neighbors. For Week 1 and Week 2, one of the clusters is distinctly split into two non-connected locations (see **??**). For this particular cluster, a different polygon was created for each of the two different locations and considered separately.

Once we create the polygons for each week, we can find the intersection of polygons for different weeks to define spatio-temporal neighbors. The coordinates of the overlapping polygons create an intersection, where these coordinates also form a polygon. Gpids are then assigned to an intersection based on which intersection coordinates contains its first observed location of the week. All of the points within that intersection are considered to be spatio-temporal neighbors, since they are located in a similar geographic region over time. For example, if we want to interpolate missing data in Week 1, would first need to find its spatio-temporal neighbors. Since it is the first week, there is no previous week information, so we can only use Week 2 to find neighbors. Next, we identify the coordinates for the intersecting polygons for these two weeks. Then, the gpids located in each intersecting polygon are found and assigned to that intersection. Gpids located in the same intersection polygon are spatio-temporal neighbors and will be used to create a model for interpolation in each of the intersecting polygons. **??** shows this process with (a) and (b) showing the polygons of the clusters for Week 1 and Week 2 respectively, and (c) showing the intersections of the polygons. Some of the intersection polygons are not shown due to their being duplicate intersections (caused by overlapping polygons), but each gpid is only assigned to one intersection. If a gpid is not found in an intersection, it is then removed from the data during this process, which is a potential area for improvement. Week 3 spatio-temporal neighbors can be found using a similar process, using just Week 2, as this was the last week in the data set. Creating the intersecting polygons for Week 2 involved the intersection of it's polygons from both Week 1 and Week 3.

In order to use this interpolation method, a spatial grid encompassing the ice sheet is created at each time. The grid is used in our created model as an estimation of the initial locations of the missing gpids, where the model will adjust this location using it's known neighbors. The size of our grid cells is 10 km by 10 km, which allows for a maximum of four gpids to be located in the cell. Additionally, this is the size of the initial grid used to track the gpids (Kwok & Cunningham 2002). The centroid of the gpids was used to estimate each grid cell, so each of the gpids located in that cell would have the same initial estimate.

2.2.2 Univariate Interpolation

Once the grid was created for initial location estimates of missing data, a univariate model was developed for both x and y. These models were creating using the GpGp package in R, which uses the Vecchia's Approximation for a Gaussian Process (Guinness & Katzfuss 2021). Vecchia (1988) developed this approximation for a Gaussian Process, as generally likelihood methods for the covariance parameter estimates of the Gaussian Process are computationally unfeasible. This method approximates the Gaussian Process by writing the joint density as a product of conditional distributions, where only a subset of the data is used to create these conditional distributions. The subset chosen greatly affects the approximation and is formed by neighbors of the observation. Ordering the the observations by one of the coordinates determines these neighbors. In Guinness (2018), updates to the ordering process were developed. A maximum minimum based ordering is used to sort the data based on sequentially picking the next point that has the maximum minimum distance to all previously selected points. Then the number of neighbors used for the subset can be defined and found from this ordering. Additionally, Guinness (2018) introduces a grouping method where groups are determined by partitioning observations into blocks, and each block's input to the likelihood can be computed at the same time. These updates were shown to further increase the accuracy of these models and lower computation time. Further, Guinness (2021), provides an efficient method for applying Fisher's scoring to maximize the log-likelihood by developing a single pass algorithm to compute the Fisher's Information and the gradient of the Vecchia Approximation. Once again, this was done to help lower computation time. The updates made in Guinness (2018) and Guinness (2021) are implemented in the GpGp package.

In order to fit the model, we need to define:

- Y = x or y. Dimension interested in (univariate response)
- loc = The matrix of x, y, and time (t) of known data. Made of locations at the desired time (t), the day before (t-1), and the day after (t+1).
- X = Is a matrix of 1's the length of the number of observed data, also known as the design matrix.
- m_seq = A sequence of values for number of neighbors in each subset. Our models

used a sequence of 10 to min(N-1, 30). Includes N-1 in case we have a small number of observations, the model does not try to create subset in which it needs more neighbors than there are observations.

We specified the covariance function as the exponential space-time covariance function, as defined by the GpGp package documentation (Guinness & Katzfuss 2021). It is defined as

$$C(\theta) = \sigma^2 e^{-||D^{-1}(x-y)||}. \quad (2)$$

where

$$D = \begin{pmatrix} \phi + c_0 & & & & \\ & \phi + c_0 & & & \\ & & \ddots & & \\ & & & \phi + c_0 & \\ & & & & \tau + c_0 \end{pmatrix}.$$

The parameters in the covariance function are of σ^2 (variance), ϕ (spatial range), τ (temporal range), and c_0 (nugget). The spatial and temporal ranges are smoothness parameters that relate to dependence over space and time respectively. The nugget parameter is the measurement error. The output is the maximum Vecchia likelihood estimates for the mean and covariance parameters. The model created by these estimates can be used to predict the unobserved locations at the time (t). The initial grid estimates of the x and y values are used as the starting locations in the prediction function and shifted based on the estimated model parameters. Predictions for x or y (depending on the Y specified in the fitted model) are made by finding the conditional expectation of the model developed.

3 Simulation Study

3.1 Data Simulation

To test the validity of our methods, we conducted a simulation study. The data was simulated to mimic the motion of sea ice, where the movement happens in patches that are driven by external factors. Separate grids are created to simulate the observed data and the underlying process that is causing the movement. First, to create the underlying process, a fine grid is created with each cell vertex representing a point. This grid is a 30x30 equally spaced grid, which is a total of 900 points. Next, initial cluster memberships are assigned to create the patches. For simplicity, the points are assigned into two clusters, each with an equal number of points. Then, this grid is shifted seven times, to represent seven days, simulating movement in the underlying process. The movement at each time step of the grid decreases over time. This data is then used in the exponential space-time covariance function (defined in (2)), along with defined parameter values to simulate a covariance matrix. The covariance function will have different parameter values for each cluster. Additionally, the parameter values may also slightly differ for X and Y within a cluster. The covariance functions and defined mean trend is then used to simulate a Gaussian Process model of the displacement for each location on the grid at a time. Hence,

$$U_{d,c}(s,t) \sim GP(\mu_{d,c}, C_{d,c}(\theta)) \quad (4),$$

where c is the cluster (c=1 or 2), d is the dimension (d=x or y), and the parameter values for the mean ($\mu_{d,c}$) and covariance function ($C_{d,c}(\theta)$) can be found in Table 1. Thus, $U_{d,c}(s,t)$ gives the displacement, or movement, for each point on the underlying grid for each day.

After the underlying process is created, a coarse grid representing the observed data is created in similar fashion to the ice data given by satellites. This is a smaller grid that is encompassed by the underlying process. Furthermore, since it is coarser, it is made up of less points. Each point is represented by $(x_{t,j}, y_{t,j})$, where t is the time, and j is the identifier used to track the movement (like a gpid). The initial observed grid values would then be represented as $(x_{t=0,j}, y_{t=0,j})$. Movement of the observed point is determined by the value of the nearest point of the underlying process for that day, determined by Euclidean Distance, to the observed point. Hence,

$$(x_{t,j}, y_{t,j}) = (U^X_{t-1,c,g}, U^Y_{t-1,c,g}) + (x_{t-1,j}, y_{t-1,j}) \quad (5),$$

where $U^d_{t,c,g}$ is the underlying process for dimension d (d=X,Y), cluster c (c=1,2), at time t-1 (t=1,...,7), for grid value g, which is the closest grid location of the underling process to $(x_{t-1,j}, y_{t-1,j})$. This process is continued until t=7 in order to get a final simulated data of a week's worth of data. Fig. 3 shows the initial grid locations for the underlying

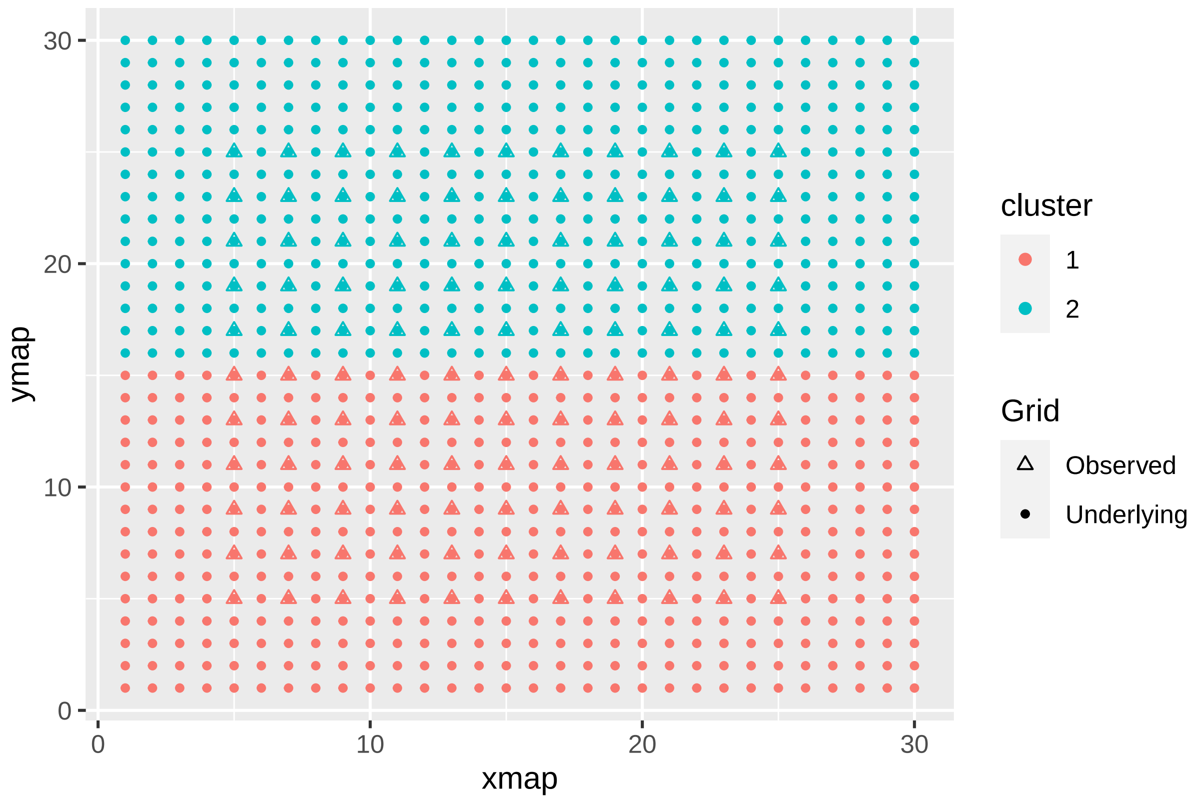

Figure 3: Underlying Process Grid and Observed Grid together, where the true cluster is also given.

process and observed data together. It also shows the true cluster membership for each grid location.

In order to evaluate our clustering and interpolation method, three different data sets were simulated in this manner with slightly different parameter values, which can be found in Table 1. A plot of the trajectories for each simulation can be found in Fig. 4. In Simulation 1 and Simulation 3, the trajectories generally start mostly linear, with curvature towards the end of the week. In Simulation 2, more curvature happens earlier in the week, with more gradual curves as the week progresses.

3.2 Clustering Method

Now, our proposed spatio-temporal clustering method, using a bounding box, is performed on each of the simulated datasets. Since the true number of clusters is two, two is used

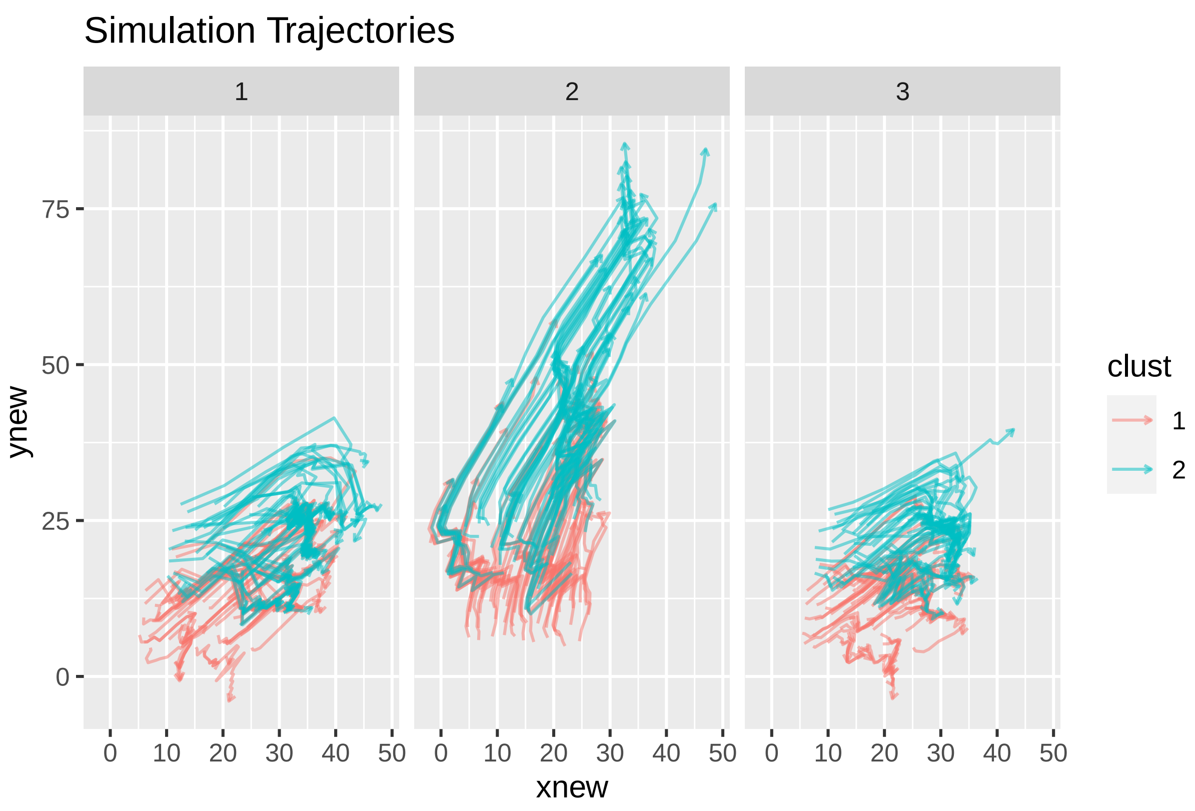

Figure 4: Trajectory Plots for each Simulated Data Set

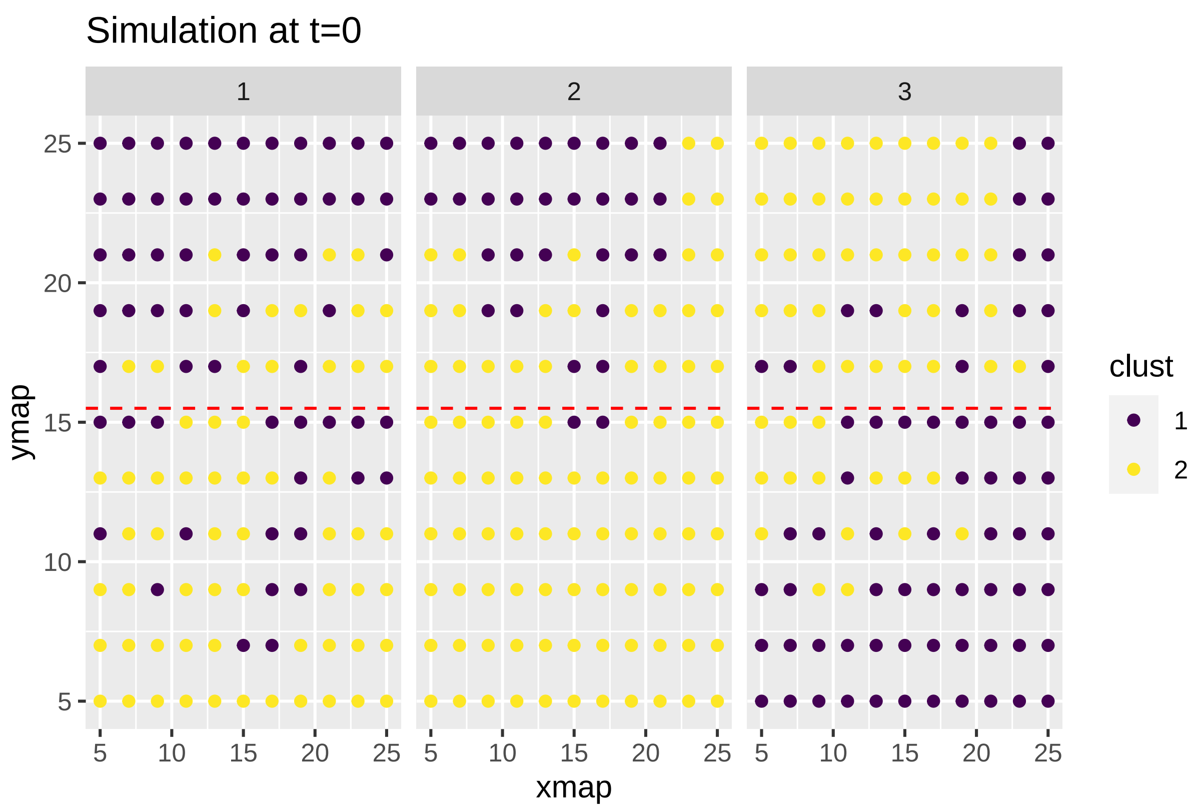

Figure 5: Clusterings of Observed Data on Original Grid

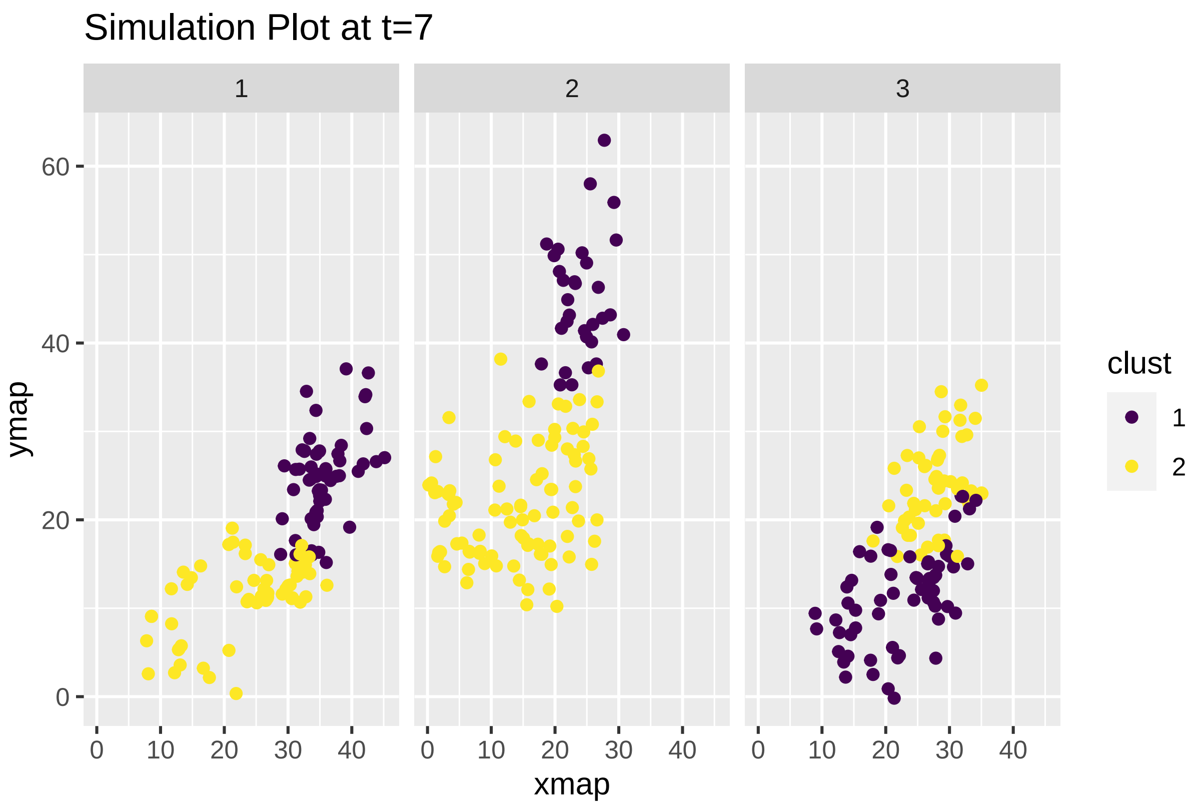

Figure 6: Clusterings of Observed Data on Final Day of Data Set

Table 1: Parameters Y for Underlying Process for Each Cluster

	X					Y				
Cluster	σ_x^2	$\phi_{x,s}$	τ_x	$Nugget_x$	μ_x	σ_y^2	$\phi_{y,s}$	τ_y	$Nugget_y$	μ_y
Simulation 1										
1	5	5	5	0	2.0	5	5	5	0	1.5
2	40	60	10	0	10.0	80	50	10	0	6.0
Simulation 2										
1	1	10	5	0	0.5	2	10	7	0	2.0
2	20	20	2	0	2.0	25	20	3	0	4.0
Simulation 3										
1	2	10	5	0	1.5	2	10	5	0	1.0
2	20	20	10	0	6.0	20	30	10	0	3.0

as the k input in the k-means clustering algorithm. The results are shown at two different time points. First, the k-means determined clusters is visualized on the initial grid to visualize how well our method performed when can see the true clusterings (see Fig. 5). The cut-off for the initial assigned cluster groupings is given by the red-dashed line. In this figure, a majority of the points seem to be clustered correctly, however, there are a number of missclassified points in each of the simulations. In Simulation 1, most of the missclassifications seem to be along the border, but do seem to increase from left to right. Most of cluster 2 is clustered correctly in Simulation 2, with the majority of missclassifications in cluster 1 happening along the edges and boundary. Similarly, in Simulation 3 most of the missclassifications are near the border, with a handful along the right edge in cluster 2. If there is a missclassified point that is surrounded by points correctly classified, this would be considered a part of the same cluster of its neighbors. This is due to the desire to have contiguous clusters and we are also primarily interested in the cluster boundaries. In the future, a post-processing filter to address these anomalies can be developed.

Table 2: RMSE for Interpolation Methods

Intersection		Linear		No Intersection	
X_{int}	Y_{int}	X_{lin}	Y_{lin}	X_{noint}	Y_{noint}
Simulation 1					
1.495585	1.516627	1.0420551	1.2255967	1.437696	1.294731
Simulation 2					
1.628306	1.579724	1.4546991	1.5396450	1.473934	1.488392
Simulation 3					
1.342212	1.337799	0.9500911	0.9206641	1.457596	1.489168

The second time point where the clusters are visualized is on the last day of the week in order to see if the clusters determined by the bounding box can distinguish movement over time. In Fig. 6, there are distinguishable boundaries between clusters for each of the simulations. Simulation 3 has a little overlap along the boundary, but can still distinguish between the two clusters. Thus, by clustering using the movement features of a trajectory, we are able to distinguish the differences in the movement patterns of the data, where the cluster boundaries to define the boundaries between the movement patterns. Some of the missclassifications on the original grid may be due to the fact that the value for the underlying process that was added to get the new location was determined by the closest grid cell of the underlying process by Euclidean distance. If the point eventually becomes closer to the other cluster, meaning that cluster's underlying process values are being added to cause the movement, it will eventually start to move like it. So if it spends more time moving like cluster 1 than cluster 2, as an example, it will be most likely be classified as cluster 1, even if that was not what it was initially assigned.

3.3 Interpolation Method

The above simulations are also used to check the performance of our spatio-temporal interpolation model. For each of the simulations, another week of data is similarly created and clustered. Polygons for the two clusters are created. Then, the intersection polygons of the weeks polygons are found. Once again, an intersections represent the spatio-temporal neighbors of the data within that intersection polygon. Next, 10% of the data for the first week are randomly assigned to be missing. Then a univariate model for x and y are developed in order to obtain our estimates of the missing locations. Additionally, in order to have a baseline to compare our method to, we also ran linear interpolation on the same data. Linear interpolation estimates an objects unknown location along a straight-line path between two known locations. It has been shown to work well for trajectories that follow a straight-line path and have a lot of sampled points. On the other hand, it tends to not work as well when a trajectory follows a more curved path or is not heavily sampled (Wentz et al. 2003, Guo et al. 2021). A third method was used for comparison that is similar to our intersection method. Here, instead of running the model inside an intersection, the intersections were ignored, and a model was developed using all known points for t-1, t, and t+1 (this essentially ignores the nonstationarity aspect of our data). The Root Mean Square Error (RMSE) for each of the simulations and interpolation methods can be found in Table 2.

Our interpolation method never seems to outperform linear interpolation. However, outside of Simulation 3, the results are not very different. In Simulation 3, running the model for each intersection performs better than the overall model. Further, since the clusters were created with different movements, in order to see if there are different areas where our method may perform better, Table 3 breaks out the RMSE calculations by cluster. In this table, we see similar results. For Simulation 1, linear interpolation always performs the best. In Simulation 2, our method outperforms linear interpolation for Y in cluster 2. However, the model using all the data performs the best here. Simulation 3 is similar to Simulation 1. Nonetheless, linear interpolation performing the best may not be surprising, as there are long periods of linear movement in each of the simulations, so the results may be dependent on what points where randomly removed. Additionally, linear interpolation can not estimate the first or last point of a trajectory, so there are fewer predicted locations used to calculate the RMSE. Thus, these simulations show that linear interpolation may generally outperform our method, but our method may have some

Table 3: RMSE for Interpolation Methods by cluster

	Intersection		Linear		No Intersection	
Cluster	X_{int}	Y_{int}	X_{lin}	Y_{lin}	X_{noint}	Y_{noint}
Simulation 1						
1	1.382578	1.555438	0.8154561	1.1625768	1.562165	1.290407
2	1.572680	1.487765	1.1881687	1.2722442	1.336760	1.297965
Simulation 2						
1	1.699892	1.653159	1.3158771	1.3290263	1.657531	1.596003
2	1.583802	1.533976	1.5034563	1.6115459	1.407423	1.450749
Simulation 3						
1	1.317954	1.346376	1.1561450	1.0959499	1.433968	1.404708
2	1.353094	1.333882	0.6469916	0.6733144	1.484391	1.581026

promise with curved data that may not be highly sampled. As an example, in $Y_{s2,c2}$, the data is more spread out and curved than the others. The other simulations may have curves, but samples are taken closer together, or the data may be spread out but mostly is linear. The results on if creating the intersections is necessary is mixed. However, these results may be impacted by some of the intersection not containing much data, which may impact model development. Through the simulations, we also found that having a fine enough grid for estimates of the missing data as starting values in the prediction function is key. If the grid is too coarse, this can impact the accuracy of the estimations.

A benefit of using a model-based approach to interpolate missing locations is that we are able to determine the uncertainty of the estimate. In order to do so, conditional draws of the unobserved values given the observed values can be used to quantify the uncertainty. This is accomplished by exploiting an advantage of Vecchia's Approximation that approximate draws from a Gaussian Process model can be made through the inverse Cholesky Factor (Guinness 2018). Therefore, for each model, 30 simulations of predictions are done and used to calculate the standard deviation. These can be used to create an interval of our

Table 4: Proportion of Prediction interval containing observed and Average Standard Deviation of Estimates for Simulated Data

Simulation	Proportion of X	Avg SD of X	Proportion of Y	Avg SD of Y
1	0.2807018	0.3819895	0.2982456	0.3511526
2	0.1875000	0.4158057	0.2500000	0.4495713
3	0.2916667	0.3715751	0.3750000	0.3841953

Table 5: Proportion of Interval Above/Under Actual Value

Simulation	X		Y	
Simulation	Above Interval	Under Interval	Above Interval	Under Interval
1	0.351	0.368	0.3421	0.281
2	0.313	0.500	0.5310	0.219
3	0.250	0.458	0.2710	0.354

Table 6: Proportion of Prediction interval containing observed by Cluster for Simulated Data

	Simulation 1		Simulation 2		Simulation 3	
Cluster	X_{s1}	Y_{s1}	X_{s2}	Y_{s2}	X_{s3}	Y_{s3}
1	0.2917	0.2083	0.3333	0.4167	0.0667	0.2667
2	0.2727	0.3636	0.1000	0.1500	0.3939	0.4242

estimates. The intervals are found by $\hat{x} \pm (2 * sd_x)$ and $\hat{y} \pm (2 * sd_y)$, where $\hat{x}$ and $\hat{y}$ are the predictions, and sd_x and sd_y are the standard deviations determined by the conditional simulations. Next, found the proportion of these intervals that contain the true value. For each simulation, the proportions can be found in Table 4, along with the average standard deviation values. The proportions in this table are low, mainly in the 20%-30% range. Like the RMSE values, the proportions can be separated by cluster, which are found in Table 6. These values are also low, with the x values for cluster 1 in Simulation 3 having almost no intervals that contain the actual value. Finally, Table 5 shows the proportion of intervals that overestimate or underestimate the actual value. For Simulations 2 and 3, the x intervals underestimate, whereas the Y intervals are an overestimate. For Simulation 1, the proportion of overestimates is slightly higher than the underestimates for Y-values, but almost identical for x.

4 Results

References

Badgley, F. (1961), Heat balance at the surface of the arctic ocean, *in* 'Proc 29th Annual Western 528 Snow Conference Spokane', pp. 101–104.

Bouillon, S. & Rampal, P. (2015), 'On producing sea ice deformation dataset from sar-derived sea ice motion', *The Cryosphere* **9**, 663–673.
URL: *www.the-cryosphere.net/9/663/2015/*

Guinness, J. (2018), 'Permutation and grouping methods for sharpening gaussian process approximations', *Technometrics* **60**(4), 415–429.

Guinness, J. (2021), 'Gaussian process learning via fisher scoring of vecchia's approximation', *Statistics and Computing* **31**(25).

Guinness, J. & Katzfuss, M. (2021), *GpGp: fast Gaussian process computation using Vecchia's approximation.* R package version 0.4.0.
URL: *https://cran.r-project.org/web/packages/GpGp*

Guo, S., Mou, J., Chen, L. & Chen, P. (2021), 'Improved kinematic interpolation for AIS trajectory reconstruction', *Ocean Engineering* **234**, 109256.

Hutter, N., Zampieri, L. & Losch, M. (2019), 'Leads and ridges in arctic sea ice from rgps data and a new tracking algorithm', *The Cryosphere* **13**(2), 627–645.

Key, J., Stone, R., Maslanik, J. & Ellefsen, E. (1993), 'The detectability of sea-ice leads in satellite data as a function of atmospheric conditions and measurement scale', *Annals of Glaciology* **17**, 227–232.

Kodinariya, T. & Makwana, P. (2013), 'Review on determining of cluster in k-means clustering', *International Journal of Advance Research in Computer Science and Management Studies* **1**, 90–95.

Kort, E., Wofsy, S. & Daube, B. e. a. (2012), 'Atmospheric observations of arctic ocean methane emissions up to 82 north', *Nature Geosciences* **5**, 318—321.

Kwok, R. & Cunningham, G. F. (2002), 'Seasonal ice area and volume production of the arctic ocean: November 1996 through april 1997', *Journal of Geophysical Research: Oceans* **107**(C10), SHE 12–1–SHE 12–17.

Lindsay, R. & Stern, H. (2003), 'The radarsat geophysical processor system: Quality of sea ice trajectory and deformation estimates', *Journal of Atomspheric and Oceanic Technology* **20**(9), 1333–1347.

Peterson, K. & Sulsky, D. (2011), Evaluating sea ice deformation in the beaufort sea using a kinematic crack algorithm with rgps data, *in* 'Remote Sensing of the Changing Oceans', Springer, Berlin, Heidelberg.

Schreyer, H. L., Sulsky, D. L., Munday, L. B., Coon, M. D. & Kwok, R. (2006), 'Elastic-decohesive constitutive model for sea ice', *Journal of Geophysical Research: Oceans* **111**(C11).

Steinley, D. (2006), 'K-means clustering: A half-century synthesis', *British Journal of Mathematical and Statistical Psychology* **59**(1), 1–34.

Vecchia, A. V. (1988), ‘Estimation and model identification for continuous spatial processes’, *Journal of the Royal Statistical Society: Series B (Methodological)* **50**(2), 297–312.

Wentz, E. A., Campbell, A. F. & Houston, R. (2003), ‘A comparison of two methods to create tracks of moving objects: linear weighted distance and constrained random walk’, *International Journal of Geographical Information Science* **17**(7), 623–645.