

ABSTRACT

SHIPP, ELENA. Data Fusion for Spatial Point Processes. (Under the direction of Dr. Erin Schliep).

The use of data fusion as a modeling technique has become increasingly prevalent in recent years. There has been great consideration of the benefits of data fusion modeling in the context of spatial and environmental statistics. Leveraging knowledge from multiple data sources is useful in a spatial statistics setting which may incorporate multiple types of data such as, geostatistical, areal, or point patterns. We explore the use of data fusion modeling through a Bayesian framework within the context of spatial point processes. We demonstrate these concepts through a simulation study and a real data application which studies the calling behavior of the North Atlantic Right Whale in Cape Cod Bay, Massachusetts.

Data Fusion for Spatial Point Processes

by Elena Shipp

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the requirement for the degree of
Master of Science

Statistics

Raleigh, North Carolina
2026

APPROVED BY:

Erin Schliep
Chair of Advisory Committee

Jonathan Duggins

Emily Griffith

BIOGRAPHY

Elena Grace Shipp was raised in Chicago, Illinois. She earned a Bachelor of Arts in Chemistry, with a minor in Science Communications, from North Carolina State University in 2024, graduating magna cum laude. That same year, she began pursuing a Master of Science in Statistics. Upon graduation, she plans to begin her career in industry.

TABLE OF CONTENTS

1	Introduction	1
2	Model	5
3	Simulation Study	8
3.1	Simulation Model Specifications	8
3.2	MCMC Implementation	11
3.3	Simulation Results	12
4	Real Data Application	19
4.1	Cape Cod Bay, Massachusetts	19
4.2	Data Fusion Approach	20
4.3	Application Results	21
5	Discussion	25
6	Appendix	27
6.1	A.1. Simulation Code	27

LIST OF TABLES

List of Tables

1	Expected counts, 95% credible intervals, and RMSE for the data fusion model and the single source models.	16
2	Empirical coverage estimates for each parameter in the data fusion model estimated from 100 iterations of the LGCP using a single covariate.	18
3	Expected counts and 95% credible intervals for each model.	24

LIST OF FIGURES

List of Figures

1	Visualizations of the simulated spatial covariate used and an example of a latent Gaussian field. (left) Surface of simulated spatial covariate, $X(\mathbf{s})$. (right) Surface of a possible latent Gaussian field that could be used as $z(\mathbf{s})$.	9
2	Visualizations of domains used for source 1 and source 2 point pattern realizations. (left) Domain for visualization of source 1, assumed to be the more accurate source. (right) Domain for visualization of source 2, defined as the less accurate source.	10
3	Simulated realizations of source 1 and source 2 data	12
4	True intensity surface and posterior mean of the intensity function obtained from fitting the data fusion model to the two sources.	13
5	Single source model intensity surfaces	14
6	Differences between true and posterior mean estimate of the log-intensity surfaces for each of the models fit. (top) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to both source 1 and source 2 data. (bottom left) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to only source 1 data. (bottom right) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to only source 2 data.	15
7	Posterior mean estimate of calibration parameter surface, obtained from fitting the data fusion model to the two sources.	17
8	Automated and manual observations of NARW calls in Cape Cod Bay. . . .	20

9	Posterior mean intensity surface obtained using data fusion model of manual and automated data sources.	22
10	Posterior mean intensity surface of single source models. (left) Model fit to only manual data (right) Model fit to only automated data.	22
11	Posterior mean of calibration parameter, $g(\mathbf{s})$	23

AUTHORSHIP STATEMENT

I, Elena Shipp, am the sole author of this thesis and take responsibility for its content.

Use of generative artificial intelligence: Generative AI was not utilized at any point in the development of this thesis.

1 Introduction

The use and study of data fusion modeling has become a rapidly growing topic of analytical interest in recent years. Data fusion has been studied through multiple lenses, such as data melding (Poole and Raftery, 2000) and data integration (Ziegler and Dittrich, 2007). Data melding and integration are techniques used to combine diverse data into a consolidated set with a main objective of creating a more usable and streamlined view of the data for analysis. Data fusion is a technique that combines multiple sources of information to achieve conclusions and make decisions with greater accuracy and specification than would be created from individual sources. The motivation for using data fusion stems from the inherent limitations of single source data. Because many data sources give significant information in one aspect yet may neglect important knowledge in another, the use of multiple sources can be incredibly useful. Although this technique is used mainly to combine the information from different types of sensor data, it can be extrapolated to various other fields such as environmental studies, defense and security operations, and many additional domains (Møller et al., 1998). In general, data fusion is used to improve four main tenets of a system: certainty, accuracy, representation, and completeness (Mitchell, 2012). As decisions in science, policy, and industry increasingly rely on data-driven insights, leveraging richer and more integrated sources of information has become essential.

One field in which data fusion is particularly relevant is environmental statistics, where data are often collected over space and time through many different sampling strategies (Banerjee et al., 2014). In practice, spatial datasets often differ in ways that complicate direct comparison, such as temporal misalignment, differences in measurement bias, mismatched spatial resolutions, and variability in sampling mechanisms. The use of data fusion remedies the issues introduced by single datasets and allows the true shared spatial process to be extrapolated despite a possible lack of information due to data source

misspecification (Gelfand and Schliep, 2025).

Spatial statistics can be broadly categorized by three main branches: geostatistics, areal data, and point patterns. Geostatistics focuses on measurements of spatially continuous processes that are collected at specific locations, which are then used to make inferences and predictions at unsampled sites. Areal data analysis utilizes observations made at regularly and irregularly spaced intervals, such as grid cells or geographic regions (e.g. counties), respectively, to make inferences based on the underlying spatial process. Point pattern analyses study spatial point processes in which the variable of interest is the location of events themselves (Cressie, 1993). The goal of regression analysis on point patterns is to define the intensity function of the underlying spatial process as a function of covariates, which allows for an intensity surface to be established over the studies region of interest. It is important to note that geostatistical and areal data fusion is the most common, whereas the use of data fusion techniques for spatial point processes is much less studied.

Data fusion can be especially valuable for spatial point processes when multiple data sets provide complementary views of the same underlying spatial phenomenon. The use of multiple point patterns to describe one feature may aid in attaining inferences with greater accuracy and completeness due to the shared underlying intensity surface.

The general notation for a spatial point pattern is:

$$\{\lambda(\mathbf{s}) : \mathbf{s} \in D\}$$

where $\lambda(\mathbf{s}) \geq 0$ is the intensity of the spatial process at a specific location \mathbf{s} . The location \mathbf{s} is located within the domain of interest D . Generally, spatial point patterns exist within 2-dimensional Euclidean space, but theoretically can be used in higher dimensions (Cressie, 1993). The domain of interest for the rest of the study will be assumed to

be in 2-dimensional Euclidean space such that $D \subset \mathbb{R}^2$.

There are many model types for spatial point processes that are used to describe natural phenomena. A widely used model for spatial point processes is the Cox process due to its ability to model clustering. A key feature of the Cox process is its "doubly stochastic" nature which allows complex structures to be modeled more realistically and completely. These processes are modeled as a nonhomogeneous Poisson process with an added random intensity measure (Bernardo et al., 1998). A specific example of a Cox process is the log-Gaussian Cox Process in which the logarithm of the intensity function itself is a Gaussian process (Møller et al., 1998). Such a process can be written as:

$$\lambda(\mathbf{s}) = \exp(\mathbf{X}(\mathbf{s})'\boldsymbol{\beta} + z(\mathbf{s})) \quad (1)$$

where $\mathbf{X}(\mathbf{s})$ represents a possible vector of spatially indexed covariates, $\boldsymbol{\beta}$ is a vector of regression coefficients, and $z(\mathbf{s})$ is a latent Gaussian random field.

There are many classification types for data fusion, often based upon the input and output of the technique. A commonly utilized framework is Dasarathy's data fusion classification. There are five categorizations in Dasarathy's hierarchical framework which include; Data In - Data Fusion Out (DAI-DAO), Data In - Feature Out (DAI-FEO), Feature In - Feature Out (FEI-FEO), Feature In - Decision Out (FEI-DEO), and Decision In - Decision Out (DEI-DEO) (Dasarathy, 1997).

Of the five categories previously described in Dasarathy's system, the Data-In-Feature-Out (DAI-FEO) category is particularly useful for modeling point processes from multiple data sources. In this category, raw data from multiple sources are modeled jointly assuming a shared intensity function to extract particular features of the process of interest (Castanedo, 2013). In this case, both data sources are used to inform $\lambda(\mathbf{s})$ in Eq. 1 above.

Modeling spatial processes within a Bayesian framework has become increasingly popular in recent years due to the ability to fully define posterior distributions based on prior information and observed data in tandem (Banerjee et al., 2014). The Bayesian framework offers clear advantages, as it provides an entire posterior distribution for the unknown parameters and latent processes rather than relying on a single point estimate and its approximate variance. This allows uncertainty to be quantified more completely and coherently (Gelfand and Schliep, 2018). Bayesian inference is especially useful in data fusion contexts as they are naturally specified within a hierarchical model framework where each data source is conditionally independent given the shared intensity function.

Gelfand and Schliep (2018) describe four key modeling concepts fundamental to hierarchical models: data with a complex structure, heterogeneity, dependent data, and contextuality. Spatial modeling exhibits increased complexity due to both first- and second-order variability. First-order effects relate to the large-scale characteristics of spatial structure specified through the mean. The second-order effects denote the spatial dependence of the process, where things in close proximity tend to be more similar than things far apart (Schliep, 2025). Model complexity is substantially increased when multiple data sources are incorporated. Hierarchical models accommodate heterogeneity of variance by allowing variance structures to be nested, which is vastly important in capturing the variability across space and data sources. Spatial data can be subject to issues of bias due to dependency between spatial structures. Hierarchical modeling also allows for scaling to be made for the appropriate coefficients based upon the nested framework, which is detrimental to the intersection of both spatial statistics and data fusion.

In combination, the ideas previously described motivate the need to develop a systematic framework to utilize multiple data sets in the context of spatial point processes. Through simulation studies and real-data applications, we demonstrate how data fusion can improve inference, reduce uncertainty, and provide a more accurate representation of the

true spatial process than either individual dataset alone.

We apply our modeling framework to study the whale calling behavior of North Atlantic Right Whales (NARW) in Cape Cod Bay, Massachusetts. Two sources of data, both collected on February 19, 2009, with varying levels of accuracy and variation will be used to create a fusion model indicative of the calling behavior and spatial distribution of the whales. Analysis on accuracy and variability of the posterior distributions will be done to compare the fusion model to the single source models.

In what follows, we present the model framework, a simulation study, the real-data application, and a discussion of our findings. Section 2 will define the spatial model used throughout the entirety of the study as well as its defined likelihood function. Section 3 will elaborate on the model specifications, including details of modeling in the Bayesian framework. In this section, we also present a simulation study and the results. Section 4 demonstrates the use of data fusion for spatial point processes in a real world context, specifically for the North Atlantic Right Whales. Lastly, section 5 synthesizes the conclusions from each of the previous sections and discusses next steps in the study of data fusion for spatial point processes.

2 Model

We model the spatial point process using a log-Gaussian Cox process characterized by an intensity function, $\lambda(\mathbf{s})$, that incorporates spatial covariates, $\mathbf{X}(\mathbf{s})$, and a Gaussian latent field, $z(\mathbf{s})$ with a specified covariance matrix. The Gaussian field is assumed mean 0 with an exponential covariance function such that $cov(z(\mathbf{s}), z(\mathbf{s}')) = \sigma^2 \exp(\frac{-d(\mathbf{s}, \mathbf{s}')}{\phi_1})$. Here, σ^2 controls the spatial variance, ϕ_1 is the range parameter, and $-d(\mathbf{s}, \mathbf{s}')$ is the distance between \mathbf{s} and \mathbf{s}' .

$$\lambda(\mathbf{s}) = \exp(\mathbf{X}(\mathbf{s})'\boldsymbol{\beta} + z(\mathbf{s})). \quad (2)$$

We assume that there are two data sources each providing partial realizations of the spatial point process. Our goal is to fuse the two sources together to improve inference. The two sources are assumed to vary in accuracy, which is built into the model. Without loss of generality, assume source 1 is the more accurate of the two sources, but only covers a portion of the study region. Assume source 2 is less accurate, but maintains a realization that spans the entire study region. Source 1 and source 2 are shown as two point patterns such that $\mathbf{S}_1 = \{\mathbf{s}_{1j} : j = 1 \dots n_1\}$ and $\mathbf{S}_2 = \{\mathbf{s}_{2m} : m = 1 \dots n_2\}$.

$$\lambda_1(\mathbf{s}) = \lambda(\mathbf{s}) \quad (3)$$

The source 2 intensity function utilizes a multiplicative calibration process, $g(\mathbf{s})$, that quantifies over and under counting in the less accurate source. The calibration process is assumed to have a mean, α , with an exponential covariance function such that $cov(g(\mathbf{s}), g(\mathbf{s}')) = \tau^2 \exp(\frac{-d(\mathbf{s}, \mathbf{s}')}{\phi_2})$. τ^2 controls the spatial variance, ϕ_2 is the range parameter, and $-d(\mathbf{s}, \mathbf{s}')$ is the distance between \mathbf{s} and \mathbf{s}' . The source 2 intensity function is written as

$$\lambda_2(\mathbf{s}) = \lambda(\mathbf{s}) \cdot \exp(g(\mathbf{s})). \quad (4)$$

The general log-likelihood function of a log-Gaussian Cox process is shown below

$$l = \sum_{i=1}^n \log \lambda(\mathbf{s}_i) - \int_D \lambda(\mathbf{s}) ds \quad (5)$$

where D is the domain of interest. Because the intensity function $\lambda(\mathbf{s})$ is stochastic it, lacks

a closed-form expression. Therefore, the integration term is evaluated using numerical integration. In this case, numerical quadrature can be used to approximate the integral using a finite sum of the grid cells, rather than using analytical methods. That is, the integral can be approximated

$$\int_D \lambda(\mathbf{s}) ds \approx \sum_{k=1}^N \lambda(\mathbf{s}_k) \Delta A_k \quad (6)$$

where ΔA_k is the area of each cell used for the numerical integration, \mathbf{s}_k are the centroids of each of the cells, and N is the total number of centroids used. Therefore, the general log-likelihood contribution is calculated as follows:

$$l = \sum_{i=1}^n \log \lambda(\mathbf{s}_i) - \sum_{k=1}^N \lambda(\mathbf{s}_k) \Delta A_k \quad (7)$$

The combined log-likelihood using both the source 1 and source 2 realizations is expressed as the sum of the individual log-likelihoods for each source. This is due to the sources being conditionally independent given the intensity function. We assume the same set of quadrature points for the numerical approximation, but this can be generalized. The combined log-likelihood can be seen below.

$$l_{combined} = \sum_{j=1}^{n_1} \log \lambda_1(\mathbf{s}_{1j}) - \sum_{k=1}^N \lambda_1(\mathbf{s}_k) \Delta A_k + \sum_{m=1}^{n_2} \log \lambda_2(\mathbf{s}_{2m}) - \sum_{k=1}^N \lambda_2(\mathbf{s}_k) \Delta A_k \quad (8)$$

where n_1 and n_2 are the total number of observations in source 1 and source 2, respectively. Through the combined likelihood, posterior inference on $\lambda(\mathbf{s})$ leverages information from both data sources.

3 Simulation Study

A simulation study was conducted to assess the proposed data fusion model for two realizations of a spatial point process. A customized Markov chain Monte Carlo algorithm was developed to sample from the full conditional distributions given the two data sources.

3.1 Simulation Model Specifications

In the simulation, the true intensity function was defined as a log-Gaussian Cox Process using a single covariate. Specifically, $\lambda(\mathbf{s}) = \exp(\beta_0 + \beta_1 X(\mathbf{s}) + z(\mathbf{s}))$ where $\mathbf{s} \subset D$ and $D = [0, 10] \times [0, 10]$. In the simulation, we define the intensity as

$$\lambda(\mathbf{s}) = \exp(1.5 + 3X(\mathbf{s}) + z(\mathbf{s})) \tag{9}$$

where $X(\mathbf{s})$ was assumed to vary spatially across the region. We generated $X(\mathbf{s})$ using a Gaussian process having mean 0 and an exponential covariance function with a variance parameter of 0.2 and range parameter of 4. The Gaussian latent field, $z(\mathbf{s})$, was generated independently from a similar spatial process with an exponential covariance function in which the variance parameter $\sigma^2 = 0.1$ and the range parameter $\phi_1 = 4$. Each were simulated at a 10×10 resolution over D . The simulated spatial covariate, $X(\mathbf{s})$, and an example of a simulated latent Gaussian field, $z(\mathbf{s})$, are shown below as Figure 1.

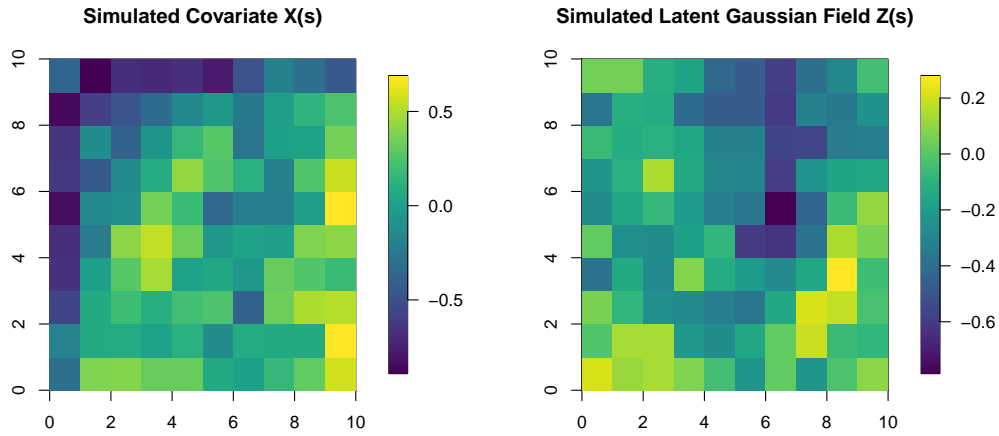


Figure 1: Visualizations of the simulated spatial covariate used and an example of a latent Gaussian field. (left) Surface of simulated spatial covariate, $X(\mathbf{s})$. (right) Surface of a possible latent Gaussian field that could be used as $z(\mathbf{s})$.

The simulated covariate within the LGCP is used to capture the possible spatial effects of a particular variable of interest. A commonly used spatial covariate is elevation. If we were to assume that the covariate used in this simulation is elevation, then the intensity surface shown in Figure 1 exemplifies lower elevations in the upper left corner and higher elevations in the center and lower right corner. The latent Gaussian field is used to account for any random spatial variability that cannot be explained by the previously defined spatial covariates. For example, the simulated latent Gaussian field shown in Figure 1 illustrates areas with lower intensities in the center and higher intensities in the lower left and right corners.

Recall we assumed source 1 is more accurate than source 2, yet source 1 is only available for a subset of D . Let $W_1 \subset D$ define the observational window for source 1. Here, we define W_1 as the subwindow containing the upper-left and lower-right quadrants of the domain. A visualization of the source 1 and source 2 domains are shown below as Figure 2.

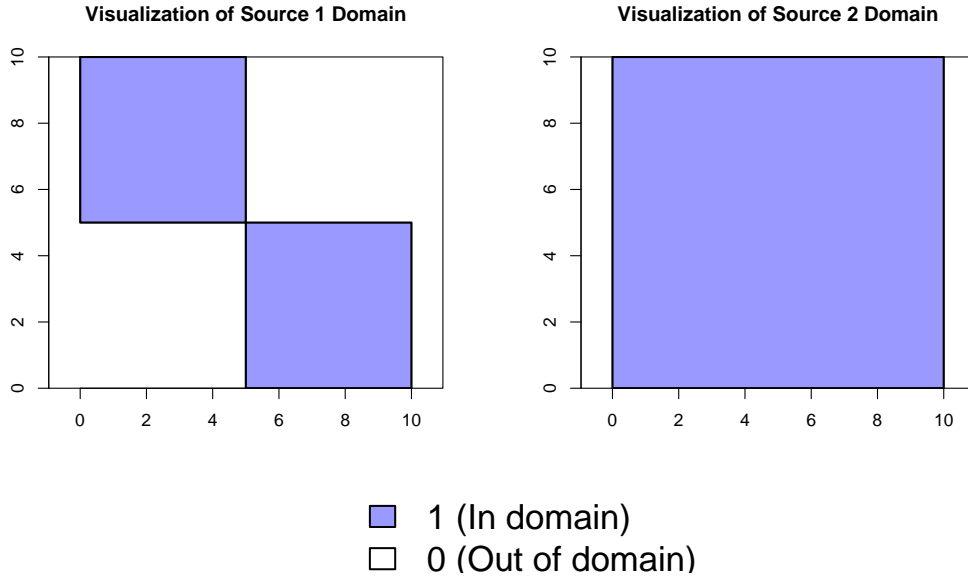


Figure 2: Visualizations of domains used for source 1 and source 2 point pattern realizations. (left) Domain for visualization of source 1, assumed to be the more accurate source. (right) Domain for visualization of source 2, defined as the less accurate source.

The multiplicative calibration parameter, $g(\mathbf{s})$, was also generated independently from a Gaussian process having a mean of -0.2 and an exponential covariance function with a variance parameter $\tau^2 = 0.4$ and range parameter $\phi_2 = 4$. Recall, $g(\mathbf{s})$ is used only in the source 2 intensity function to quantify misdetections in the less accurate source.

Then, spatial point patterns for both sources were simulated independently using their respective intensity functions

$$\lambda_1(\mathbf{s}) = \begin{cases} \exp(1.5 + 3X(\mathbf{s}) + z(\mathbf{s})), & s \in \mathcal{W}_1, \\ 0, & s \notin \mathcal{W}_1 \end{cases} \quad (10)$$

$$\lambda_2(\mathbf{s}) = \exp(1.5 + 3X(\mathbf{s}) + z(\mathbf{s}) + g(\mathbf{s})). \quad (11)$$

3.2 MCMC Implementation

A Markov chain Monte Carlo (MCMC) algorithm was implemented to obtain posterior samples of the parameters that govern the behavior of the LGCP. The parameters of interest that were extracted include the regression parameters (β_0 and β_1), the measurement error process for source 2 ($g(\mathbf{s})$), the latent Gaussian field ($z(\mathbf{s})$), the variances of the latent Gaussian field and source 2 process (σ^2 and τ^2 , respectively), and the mean of the measurement error process of source 2 (α). To complete the Bayesian algorithm, prior distributions were assigned to all unknown parameters. The priors were chosen to give either weakly informative knowledge or general assumptions about the scale and variability of each parameter.

The regression coefficients, β_0 and β_1 , were assigned weakly informative Normal priors with mean 0 and variance 100, reflecting limited prior knowledge of their magnitudes. The latent spatial field, $z(\mathbf{s})$, and the Source 2 measurement error field, $g(\mathbf{s})$, were each modeled as Gaussian processes. The corresponding variance components, σ^2 for $z(\mathbf{s})$ and τ^2 for $g(\mathbf{s})$, were assigned conjugate Inverse-Gamma(3,1) and Inverse-Gamma(2,1) priors, respectively. For computational benefits, we fixed both the spatial range parameters for $z(\mathbf{s})$ and $g(\mathbf{s})$ to 4. Finally, the mean measurement error parameter for Source 2, α , was assigned a conjugate Normal(0, 100) prior, representing uncertainty in the average bias of the Source 2 observations.

The MCMC algorithm was a hybrid Metropolis-within-Gibbs algorithm. The slope parameters, β_0 and β_1 , were jointly updated using a Metropolis-Hastings algorithm. The latent spatial field, $z(\mathbf{s})$, and the source 2 measurement error field, $g(\mathbf{s})$, were both sampled by elliptical slice sampling (Murray et al., 2010). This algorithm is specifically used for models involving multivariate Normal priors. Elliptical slice sampling offers a much more efficient and tuning-free alternative to other sampling methods, lending itself to be a computationally efficient sampler for high-dimensional structures.

The variance parameters, τ^2 and σ^2 , were both updated using Gibbs sampling, specifically the Normal-Inverse-Gamma prior assumption (Reich and Ghosh, 2019). This assumption allows the parameters to be sampled directly from a conditional posterior distribution. The mean parameter for the measurement error for source 2, α , was also updated using the Normal-Inverse-Gamma prior assumption (Reich and Ghosh, 2019). The parameter α was sampled directly from the conditional posterior distribution.

A realization of the two observed point patterns from the sources described earlier are shown in Figure 3. These observed realizations of the true point process were fitted using the data fusion model to estimate the intensity surface of the underlying spatial process.

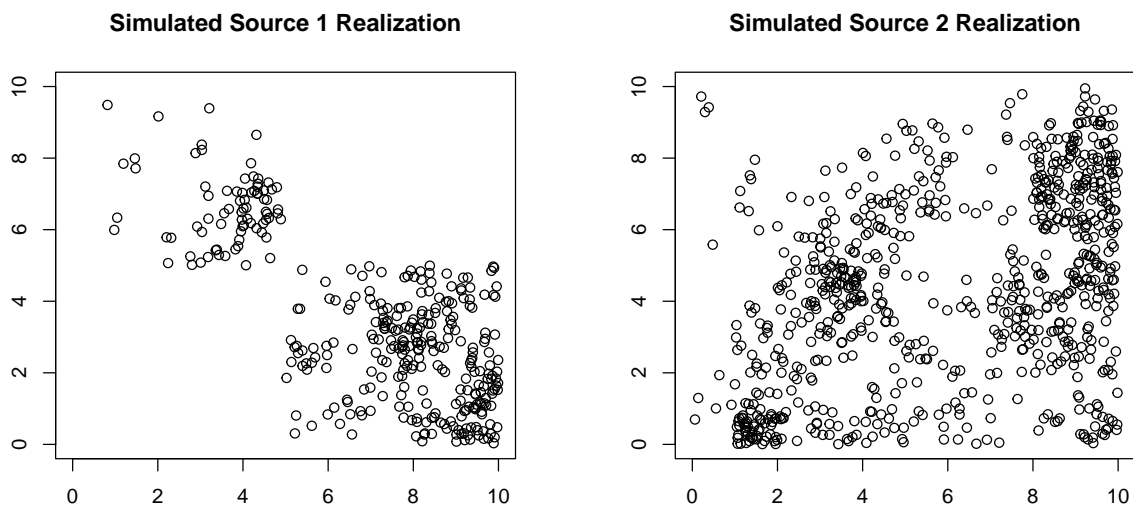


Figure 3: Simulated realizations of source 1 and source 2 data

3.3 Simulation Results

We ran the MCMC for 10,000 iterations from the joint posterior distribution for each simulation. Conclusions regarding the efficacy and accuracy of the data fusion algorithm were drawn from empirical coverage calculations and individual simulation studies.

The posterior mean intensity of the fusion model can be compared to the true intensity surface. Both intensity surfaces, the true and the one estimated from the fusion model, can be seen in Figure 4. Visually, the true intensity surface and posterior intensity surface estimate from the fusion model are very similar, showing visible evidence of the accuracy of the fusion model.

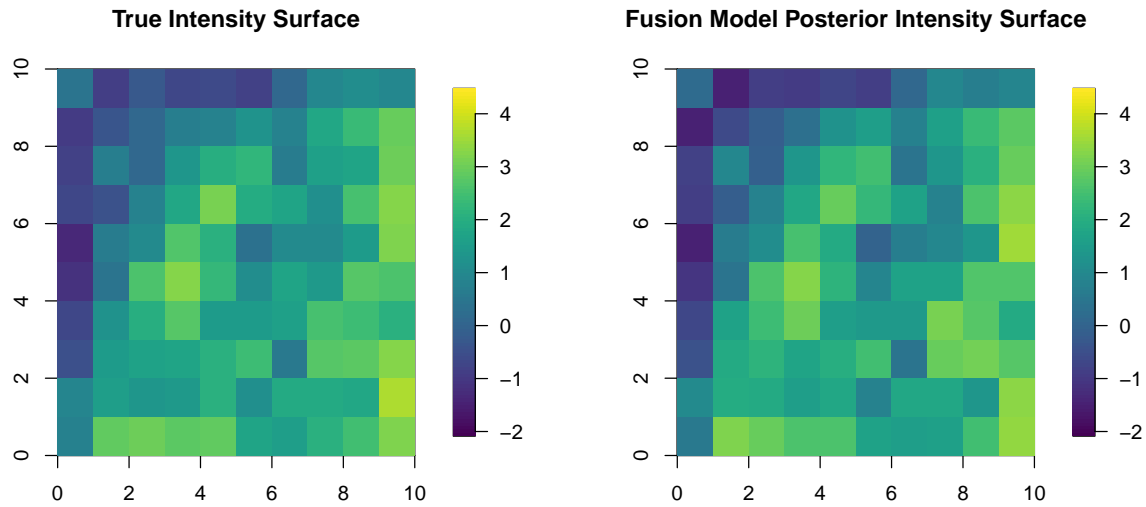


Figure 4: True intensity surface and posterior mean of the intensity function obtained from fitting the data fusion model to the two sources.

To assess the benefits of combining the sources, the fusion model intensity estimates were compared to intensity estimates of the single source models. The model was refit to each source independently to estimate their individual intensity functions, simulating a scenario in which only a single dataset was available. The intensity surface plots for the independently estimated intensity functions are shown below as Figure 5. The single source models both show that the general spatial process is captured, but areas of higher intensity are overestimated when compared to the true intensity surface. Additionally, the model fit with only source 2 data shows consistent underestimation in the upper left area of the domain.

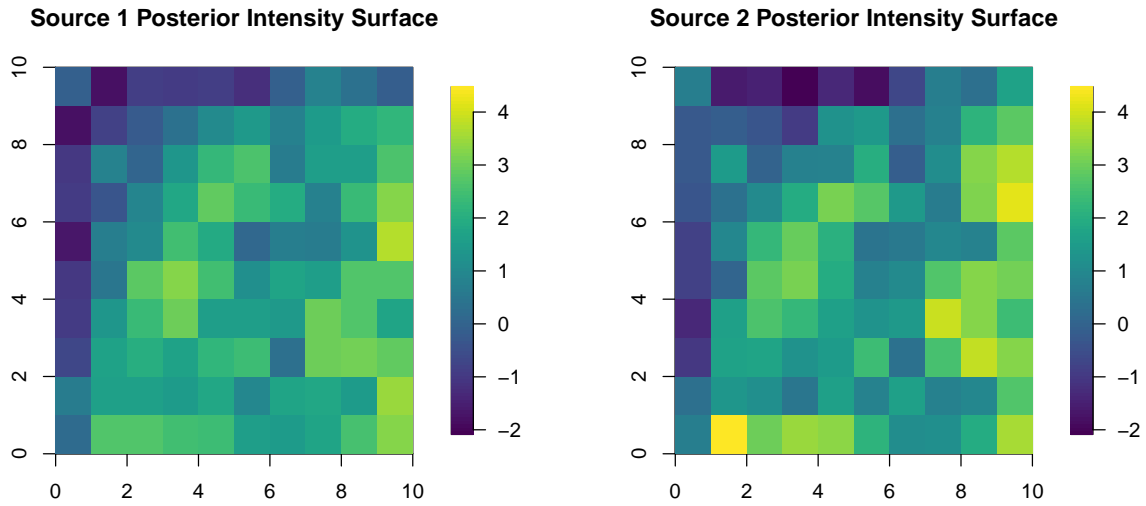


Figure 5: Single source model intensity surfaces

Additional visualizations of the difference between the true intensity surface and estimated intensity surfaces are shown below as Figure 6. The plotted differences allow for comparisons between models to be made more easily as the contrasts in color denote the extent to which each model diverges from the true intensity. It can be seen that the plot showing the difference between the true and estimated intensity surface fitted to the data fusion model contain the least differences as a majority of the colors are very light or close to white. The difference between the true and estimated intensity surface fitted to the source 1 model are very similar, but there is a clear divergence from the true intensity values in the upper left and right corners of the domain due to the bright red coloration denoting underestimation in the intensity values from the source 1 data. Lastly, the plot showing the difference between the true and estimated intensity surface fitted to the source 2 model show evidence of the most contrast due to the bright blue and red coloration, denoting that there is both over- and underestimation of the intensity function throughout the entire domain. Overall, we can conclude that the fusion model posterior intensity surface is the most similar to the true intensity surface while

the source 2 posterior intensity surface is the least accurate, or most different, from the true intensity surface.

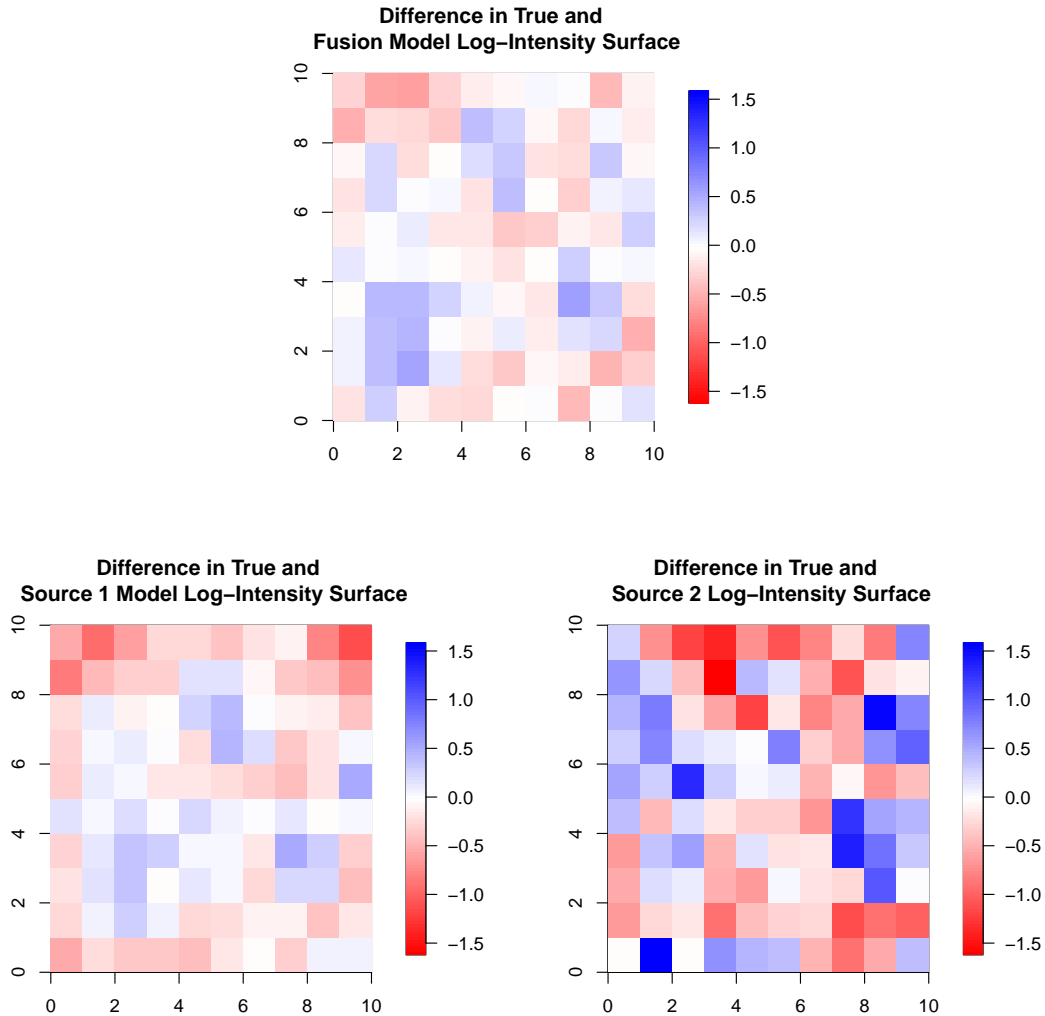


Figure 6: Differences between true and posterior mean estimate of the log-intensity surfaces for each of the models fit. (top) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to both source 1 and source 2 data. (bottom left) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to only source 1 data. (bottom right) Difference between true intensity surface and posterior mean estimate of log-intensity surface of model fit to only source 2 data.

The expected counts over the domain for each of the models, fusion, source 1, and source 2, were estimated using the posterior intensity estimates. The 95% credible intervals were calculated along with the proportional widths of each interval for comparison between sources. The proportional width of each 95% credible interval was calculated by subtracting the lower limit from the upper limit and dividing by the expected count. The standardization of the credible interval width allows for comparison between varying expected counts. The posterior expected counts are shown below in Table 1.

Model	Expected Count	95% Credible Interval	Proportional Width	RMSE
Fusion	808.19	(738.63, 883.31)	0.18	1.39
Source 1	723.94	(591.03, 890.13)	0.41	5.04
Source 2	981.72	(905.46, 1060.86)	0.16	20.74

Table 1: Expected counts, 95% credible intervals, and RMSE for the data fusion model and the single source models.

The true expected count calculated as the integral of the true intensity function over the domain, $\int_D \lambda(\mathbf{s})d\mathbf{s}$, was 774.34. Both the data fusion model and model fitted with only source 1 data captured the true expected count within their respective 95% credible intervals. However, although the true count was captured, the source 1 model had a larger proportional width of 0.41, indicating substantially more uncertainty in its estimation. The fusion model maintained a smaller proportional width of 0.18. The source 2 model significantly overestimated the expected count but had a proportional width that was smaller than both the fusion model and model fitted to source 1. In this case, accuracy was sacrificed for a slightly lower variance in estimation. We also compared the fits using RMSE between the true intensity and estimated intensity across the set of gridded locations. The fusion model RMSE calculations show that the difference between the predicted intensity values and the true intensity values were the lowest by a large margin,

therefore, providing the most accurate intensity prediction. Overall, the fusion model successfully leverages the accuracy of data source 1 and the full spatial coverage of source 2 to produce an estimate that was both accurate and consistent.

The calibration parameter, $g(\mathbf{s})$, is used to show the areas in which the less accurate source is over- or underestimating the intensity of the spatial process. Below is the posterior mean surface of the calibration parameter shown as Figure 7.

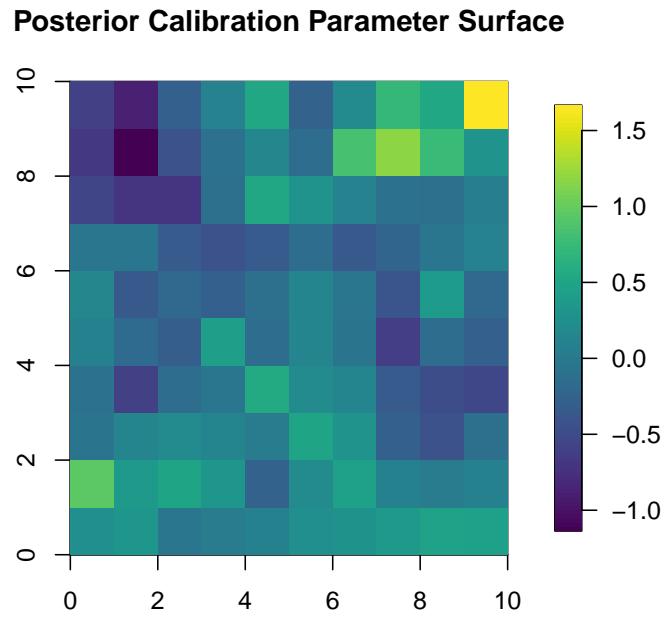


Figure 7: Posterior mean estimate of calibration parameter surface, obtained from fitting the data fusion model to the two sources.

The calibration parameter surface indicates that the upper left area of the domain was largely underestimated in the source 2 data provided. The top right area of the domain also indicates that the intensity was substantially overestimated in the source 2 data. Both of these instances show the contributions to the observed differences in accuracy and variation between the two sources.

The empirical coverage of the data fusion model was evaluated by simulating 100 iterations of the spatial point process using a single covariate. For each iteration, a 95% credible interval was generated for each of the model parameters as well as the expected count. For each simulation, it was recorded whether this interval included the true values of the underlying point process. The empirical coverage for each parameter is shown below in Table 2.

Parameter	Empirical Coverage	Bias
Expected Count	0.96	0.01
β_0	0.20	-0.75
β_1	0.90	-0.05
σ^2	0.00	-0.95
τ^2	0.89	-0.06
α	0.99	0.04

Table 2: Empirical coverage estimates for each parameter in the data fusion model estimated from 100 iterations of the LGCP using a single covariate.

The empirical coverage of the expected count was 0.96, indicating that the algorithm provides adequate coverage. It can be seen that there is a wide variety of empirical coverage between the other parameters. Their reduction in empirical coverage can be attributed to correlation that exists between model parameters. This in part, is due to the fact that the spatial Gaussian Process, $z(\mathbf{s})$ can also lead to spatial confounding, which will result in biased parameter estimates. The issues of bias in both σ^2 and β_0 could be alleviated with the use of hierarchical centering (Hofmann and Gavin, 1998). This centering method is used to account for the effects one parameter may have on another across multiple levels of a hierarchical model. Importantly, the empirical coverage of β_1 is close to the nominal level as this parameter captures the covariate effect of the spatial intensity. The expected count utilizes the composition of all parameters collectively, therefore, is most indicative

of the true empirical coverage of the fusion model. These results suggest that the data fusion algorithm reliably estimates the uncertainty in the expected counts.

4 Real Data Application

4.1 Cape Cod Bay, Massachusetts

The aquatic environment of Cape Cod Bay, Massachusetts provides an example of the importance of data fusion in environmental research. Cape Cod Bay is an essential feeding and calving ground for the endangered North Atlantic right whale (NARW). Because the population of this species is declining, researchers are eager to observe these animals in their natural environment. In many cases, multiple methods of data collections are needed to gain a comprehensive understanding of NARW behavior.

Passive acoustic monitors (PAMs) are used to record sounds occurring in the waters of Cape Cod Bay (Schliep et al., 2024). The resulting sound files from these sensors are used to geolocate whale calls based on triangulation. Trained analysts can distinguish the different sounds within the sensor files to identify the calls made by NARWs. We denote this data source as the manual data. This process is incredibly tedious and inefficient causing only a portion of the audio files to be analyzed by humans. Conversely, computer programs have been developed to automatically partition the sound files based on various sources of noise that are known to be found in the bay area. This data source is specified as the automated data. These programs can quickly and efficiently determine the locations of NARWs over the entire duration of the recordings. However, these programs are prone to have a much higher measurement error leading to misdetections and false detections. In this case, the ability to study a greater area comes at the cost of accurate information. By combining both human and computer-generated data through data fusion techniques, it is possible to gain a more accurate and comprehensive understanding of the spatial

distribution of NARWs in Cape Cod Bay (Clark et al., 2010).

4.2 Data Fusion Approach

The spatial point patterns from both data sources, the manual and automated observations, were jointly modeled to study the calling behavior of NARWs. The manual observations data frame contained 1700 whale call events and the automated observations contained 406 whale call events. Both of the sources were collected on February 19, 2009. The spatial region spans the area between 41.72413°N to 42.2118°N latitude and 70.57832°W to 69.96345°W longitude. The spatial point patterns for each source are shown in Figure 8.

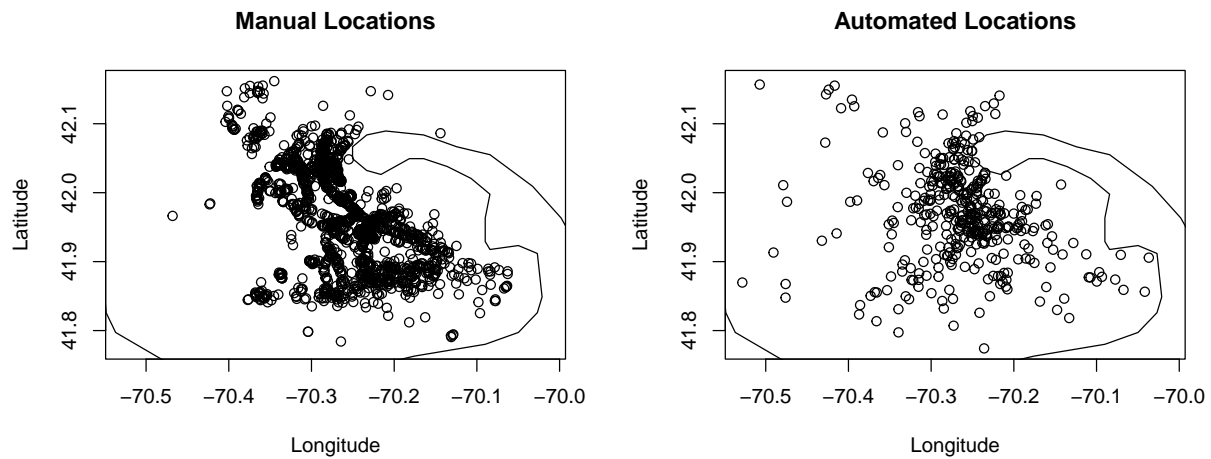


Figure 8: Automated and manual observations of NARW calls in Cape Cod Bay.

The true intensity function was modeled as a log-Gaussian Cox Process (LGCP) with only an intercept term, β_0 . The intensity function model is as follows:

$$\lambda(s) = \exp(\beta_0 + z(s)) \quad (12)$$

The regression coefficient, β_0 , was assumed to be normally distributed with a mean of

0 and a variance of 100, denoting weak prior knowledge regarding the intercept which captures the average call rate per unit space. The latent field, $z(\mathbf{s})$, was assumed to be a mean-zero Gaussian process with an exponential covariance function and a range parameter of 0.1 which corresponds to an effective range of 0.3 degrees or approximately 33 km. The variance of the latent field, σ^2 , was given an Inverse-Gamma(2,1) prior. The sensor bias field, $g(\mathbf{s})$, was also assumed to be a Gaussian process with exponential covariance function and a range parameter of 0.1. The variance of the sensor bias parameter, τ^2 , was given a Inverse-Gamma(2,1). The mean parameter of the sensor bias parameter, α , was assigned a normal prior with mean 0 and variance 100. A grid resolution of 20×20 was used in this window to generate the points for numerical integration. Only cells located in Cape Cod Bay were included. Any data points that were geolocated to a position not in Cape Code Bay were omitted. As a result, 334 quadrature points were used for the approximation of the stochastic integral.

4.3 Application Results

Using the specified priors and MCMC algorithm, we obtained 10,000 samples from the posterior distribution. Intensity surfaces were then estimated for each of the single source models, as well as using the fusion model, resulting in three estimates of the same underlying spatial process. The intensity surface estimated from the fusion model represents information from both observations of the spatial point process. The intensity surface using the fusion model is shown below as Figure 9. The intensity surface estimated only using the manual and automated data are also shown below as Figure 10.

Fusion Model Posterior Intensity Surface

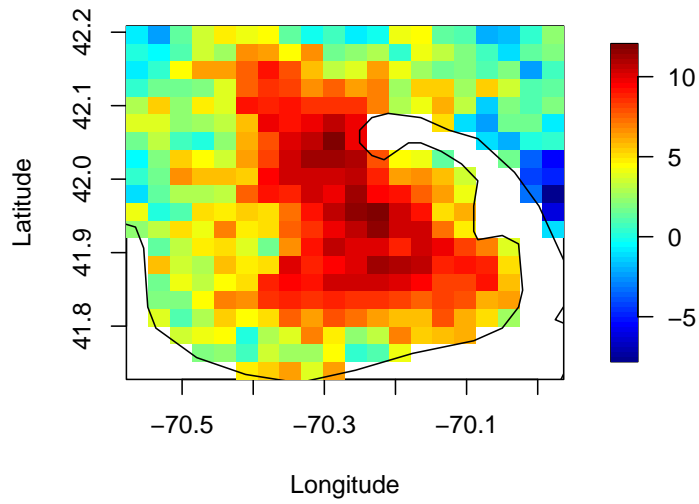


Figure 9: Posterior mean intensity surface obtained using data fusion model of manual and automated data sources.

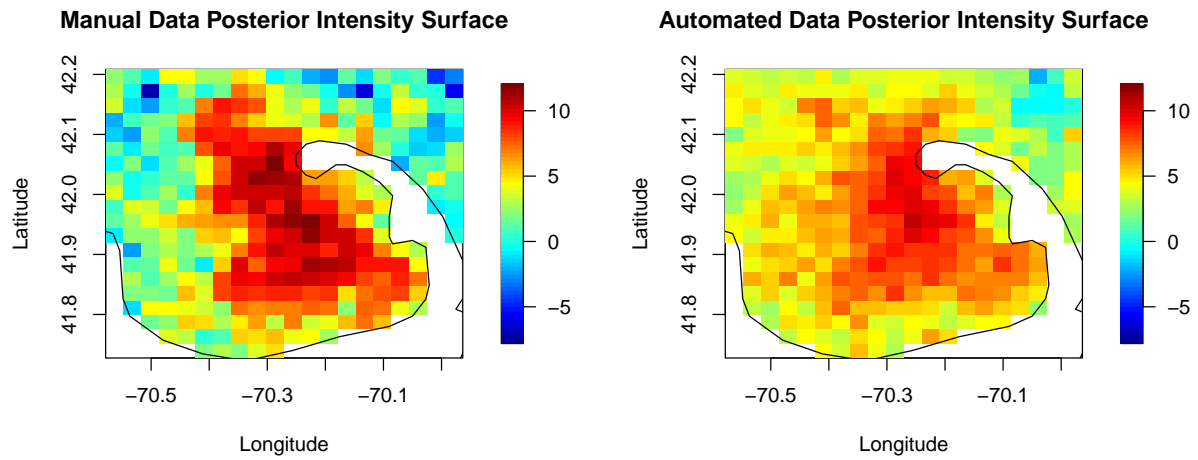


Figure 10: Posterior mean intensity surface of single source models. (left) Model fit to only manual data (right) Model fit to only automated data.

Recall that the manual data (source 1) is the more accurate data source while the automated data (source 2) is the less accurate data source. The model fit to the automated data shows the most dispersion of high intensity values across the region when compared to both the fusion model and model fit to the manual data. The fusion model shows high intensity areas in the central and eastern-central areas of Cape Cod Bay with lowest intensities shown in the north-western corner of the domain. In context, it is understandable that the lowest intensities are found in this area as there is a major shipping lane that follows this trajectory.

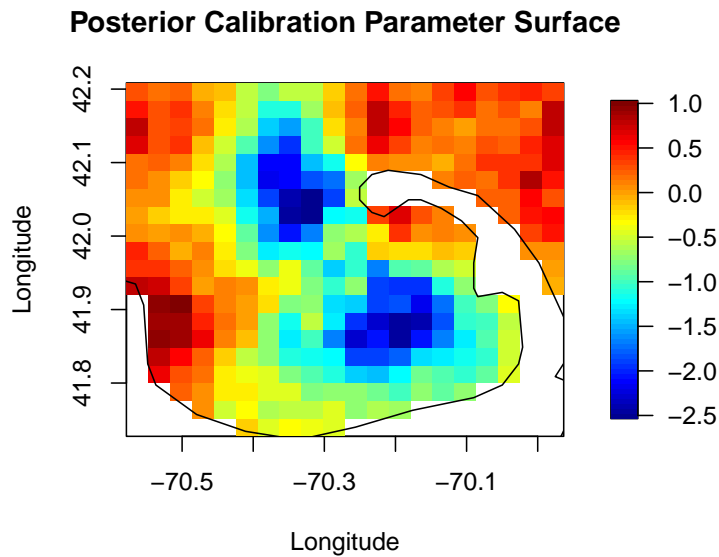


Figure 11: Posterior mean of calibration parameter, $g(\mathbf{s})$.

The calibration parameter, $g(\mathbf{s})$, is used as a correction process to link the two data sources that have differing measuring errors. This parameter specifically captures the regions in which the less accurate source, the automated source, has a misdetection or false detections. Wherever the automated data model tends to over-count or under-count events,

estimates of $g(\mathbf{s})$ will be positive or negative, respectively. We assume $g(\mathbf{s})$ varies smoothly over space, such that it captures spatially structured sensor calibration error rather than random noise. The estimated intensity surface of $g(\mathbf{s})$ is shown as Figure 11.

The posterior mean surface estimate of $g(\mathbf{s})$ shows that the main areas that tend to be over-counted by the automated data source, or source 2, are in the western and north-eastern sections of the bay. The area on the west coincides with the previously mentioned shipping lane that runs through the west coast of Cape Fear Bay. This overcounting may be a result of sound waves created by the ships that create errors in the detection of the NARW calls. The automated data collections tends to under-count in areas that have higher intensities of whale calls as seen in the central sections of the bay. Importantly, the automated data only detected about 25% of the whale calls, therefore, source 2 significantly under-counted overall. This pattern suggests that compounding noise may reduce the accuracy of automated readings in areas with higher call activity.

Based on the models fit, we can compute the expected number of calls throughout the spatial domain. The posterior mean and 95% credible intervals for each model are reported in Table 3. Not surprisingly, the expected number of points is much larger for the manual than the automated single source models.

Model	Expected Count	95% Credible Interval	Proportional Width
Fusion	1695.46	(1613.35 , 1778.39)	0.0973
Manual	1701.21	(1620.94, 1783.31)	0.0954
Automated	401.99	(326.93, 485.88)	0.395

Table 3: Expected counts and 95% credible intervals for each model.

It can be seen that the fusion model and model fit to only the manual data are comparable. The fusion model and model fit with just the manual data have very similar proportional widths. The fusion model shows that the addition of the automated data does not have

any effect when both data sources are observed over the entire domain. The automated source has the largest proportional credible interval width reflecting that it is the least accurate source. There may be a more significant decrease in the variation of the fusion model if the manual data model were to be collected over a subregion of the domain rather than the entire domain.

5 Discussion

As data collection methods continue to expand, the use of data fusion as a technique to produce more accurate and specific inference will be crucial to scientific endeavors. An area in which the fusion of multiple data sets will be vital is spatial statistics due to the variability of spatial data which often complicates direct comparison. In this study, we were able to show the importance and usefulness of data fusion in the context of spatial point patterns through both simulation and a real data application. Both the studies showcased the use of hierarchical modeling within a Bayesian framework to accurately infer the intensity function of a spatial point process.

The simulation study utilized two data sources: a source that is more accurate, but only covers a portion of the study region, and a source that is less accurate, but covers the entirety of the study region. The fusion model allowed for more accurate and less variable inferences to be made when compared to either of the single source models.

The real data application used whale call data of the North Atlantic Right Whale (NARW) in Cape Cod Bay, Massachusetts. Again, two data sources with varying levels of accuracy were used. The data in the first source was collected by trained analysts that are assumed to be more accurate. The second source data was collected by computer programs that are able to automatically partition the sound files which are assumed to be less accurate, but more efficient. It was shown that the fusion model provided consistent estimates with uncertainty that is comparable to the most accurate data source of the whale call counts

when compared to the models that fit the two data sources independently.

Continued work in the use of data fusion techniques for spatial point processes will be incredibly important in the coming years. Future work in this field could include increasing computational power and output to improve resolution, the addition of spatial covariates into the intensity function to account for more variability when modeling, and the inclusion of more data sources. For example, the extension of this data fusion technique to spatiotemporal data would allow the fusion model to account for changes in both time and space. Specifically, the addition of time as a variable of interest for the NARW study would give insight into their daily behaviors, such as how the whale calling patterns change on an hourly basis and whether the whales show preferences to certain parts of the bay based on the time of day. Additionally, adding a third data source into the fusion model for the NARW study would allow for areal or geostatistical sensor data to be incorporated into the process. Data fusion techniques within the sphere of spatiotemporal modeling has the ability to be extended to copious areas for the advancement of inference and problem solving.

6 Appendix

6.1 A.1. Simulation Code

```
# Packages
library ( spatstat )
library ( fields )
library (FastGP)
library (MASS)
library (ggplot2)
library (coda)

# True LGCP -----
# Simulate Covariate
win <- owin(xrange = c(0, 10), yrange = c(0, 10))

grid_res <- 10

cell_size <- diff(win$xrange) / grid_res

x_seq <- seq(win$xrange[1] + cell_size/2,
            win$xrange[2] - cell_size/2,
            by = cell_size )
y_seq <- seq(win$yrange[1] + cell_size/2,
            win$yrange[2] - cell_size/2,
            by = cell_size )

coords <- as.matrix(expand.grid(y_seq, x_seq))
grid_coords <- expand.grid(x = y_seq, y = x_seq)

dists <- as.matrix(dist(coords))
n <- nrow(coords)

S <- 0.2 * exp(-dists/4)

mu <- rep(0, n)

covariate <- as.vector(rcpp_rmvnorm(1,S,mu))

cov_field <- matrix(covariate - mean(covariate),
                    nrow = grid_res,
                    ncol = grid_res,
                    byrow = TRUE)
```

```

cov_field_ppp <- ppp(x = grid_coords$x,
                    y = grid_coords$y,
                    window = win,
                    marks = covariate)

#Simulate Gaussian random field

sigma_2 <- 0.10

S_z <- sigma_2 * exp(-dists/4)

z <- as.vector(rcpp_rmvnorm(1,S_z,mu))

z_mat <- matrix(z, nrow = length(x_seq), ncol = length(y_seq))

z_ppp <- ppp(x = grid_coords$x,
            y = grid_coords$y,
            window = win,
            marks = z)

# Simulate LGCP
b_0 <- 1.5
b_1 <- 3

lambda <- exp(b_0 + b_1*(cov_field) + z)
lambda_im <- im(lambda, xcol = x_seq, yrow = y_seq)

lgcp_sim <- rpoispp(lambda_im)

# Discretize using spatstat

lgcp_discretize <- pixellate(lgcp_sim, eps = 1)

# Source 1 -----

nrow <- length(lgcp_discretize$yrow)
ncol <- length(lgcp_discretize$xcol)

x_min_subwindow1 <- 0
x_max_subwindow1 <- 5
y_min_subwindow1 <- 5
y_max_subwindow1 <- 10

```

```

x_min_subwindow2 <- 5
x_max_subwindow2 <- 10
y_min_subwindow2 <- 0
y_max_subwindow2 <- 5

sub_window1 <- owin(xrange = c(x_min_subwindow1, x_max_subwindow1),
                    yrange = c(y_min_subwindow1, y_max_subwindow1))

sub_window2 <- owin(xrange = c(x_min_subwindow2, x_max_subwindow2),
                    yrange = c(y_min_subwindow2, y_max_subwindow2))

lambda_1 <- lgcp_discretize

lgcp_1_sub1 <- rpoispp(lambda_1[sub_window1])
lgcp_1_sub2 <- rpoispp(lambda_1[sub_window2])

lgcp_1 <- superimpose(lgcp_1_sub1, lgcp_1_sub2,
                      W = owin(c(0,10), c(0,10)))

# Source 2 -----

tau_2 <- 0.4
S_g <- tau_2 * exp(-dists/4)
alpha <- -0.2

g <- as.vector(rcpp_rmvnorm(1,S_g,alpha))

exp_g <- exp(g)

lambda_2 <- lgcp_discretize * exp_g

lgcp_2 <- rpoispp(lambda_2)

# Data frame creation -----

## X grid
X_grid <- as.data.frame(coords)
colnames(X_grid) <- c("x", "y")
X_grid$covariate <- covariate

## Source 1 (small var)
X_1 <- as.data.frame(lgcp_1)
nn_X_1 <- nncross(lgcp_1, cov_field_ppp)
X_1$covariate <- cov_field_ppp$marks[nn_X_1$which]

```

```

# making a mask for X-grid to be used with source 1
inside_sub1 <- with(X_grid,
  x >= x_min_subwindow1 & x <= x_max_subwindow1 &
  y >= y_min_subwindow1 & y <= y_max_subwindow1)

inside_sub2 <- with(X_grid,
  x >= x_min_subwindow2 & x <= x_max_subwindow2 &
  y >= y_min_subwindow2 & y <= y_max_subwindow2)

X_grid$mask_source1 <- inside_sub1 | inside_sub2

## Source 2 (large var)
X_2 <- as.data.frame(lgcp_2)
nn_X_2 <- nncross(lgcp_2, cov_field_ppp)
X_2$covariate <- cov_field_ppp$marks[nn_X_2$which]

# MCMC -----

## log likelihood function
loglike <- function(parameters, data) {

  idx_mask1 <- which(data$X_grid$mask_source1)

  log_lambda_points1 <- parameters$beta[1] + parameters$beta[2] *
    data$X_1$covariate + parameters$z[data$nn_index_1]
  term1 <- sum(log_lambda_points1)

  log_lambda_grid1_full <- parameters$beta[1] + parameters$beta[2] *
    data$X_grid$covariate + parameters$z

  lambda_grid1_masked <- exp(log_lambda_grid1_full[idx_mask1])
  term2 <- sum(lambda_grid1_masked * data$cell_area)

  log_lambda_points2 <- parameters$beta[1] + parameters$beta[2] *
    data$X_2$covariate + parameters$g[data$nn_index_2]
    + parameters$z[data$nn_index_2]
  term3 <- sum(log_lambda_points2)

  log_lambda_grid2 <- parameters$beta[1] + parameters$beta[2] *
    ↪ data$X_grid$covariate + parameters$g + parameters$z
  lambda_grid2 <- exp(log_lambda_grid2)
  term4 <- sum(lambda_grid2 * data$cell_area)

  likelihood <- (term1 - term2) + (term3 - term4)
  return(likelihood)}

```

```

## Updating slope estimates using Metropolis Hastings MCMC
update_betas<- function(parameters, priors, data){

  beta_cand <- rnorm(length(parameters$beta), mean = parameters$beta, sd =
    ↪ priors$beta_prop_sd)

  params_top <- list(beta = beta_cand, g = parameters$g, z = parameters$z)
  params_bottom <- list(beta = parameters$beta, g = parameters$g, z = parameters$z)

  # Posteriors
  post_top <- loglike(params_top, data) + sum(dnorm(beta_cand, mean =
    ↪ priors$beta_mean, sd = priors$beta_sd, log = TRUE))
  post_bottom <- loglike(params_bottom, data) + sum(dnorm(parameters$beta, mean
    ↪ = priors$beta_mean, sd = priors$beta_sd, log = TRUE))

  # Metropolis Hastings Ratio
  log_acceptance_ratio <- post_top - post_bottom

  if (log(runif(1)) < log_acceptance_ratio) {
    parameters$beta <- beta_cand
  }

  return(parameters) }

#Updating g – Source 2 measurement error
update_g <- function(parameters, priors, data){

  # Choosing ellipse (nu) from prior (g)
  nu <- as.vector(MASS::mvrnorm(n = 1, mu = rep(0, length(parameters$g)), Sigma =
    ↪ parameters$tau_2 * exp(-data$dists/4)))

  # Log likelihood threshold (finding log(y))

  u <- runif(1, min = 0, max = 1)

  log_y <- loglike(parameters, data) + log(u)

  # Draw an initial proposal for theta

  theta <- runif(1, min = 0, max = 2*pi)
  theta_min <- theta - 2*pi
  theta_max <- theta

```

```

repeat {
  # Calculate g'
  g_prime <- as.vector((parameters$g - parameters$alpha) *cos(theta) + nu*sin(
  ↪ theta))

  params_prime <- parameters
  params_prime$g <- g_prime + parameters$alpha

  # Shrinking bracket
  if (loglike(params_prime, data) > log_y){
    parameters$g <- g_prime + parameters$alpha
    return(parameters)
  } else {
    if(theta < 0) {
      theta_min <- theta
    } else {
      theta_max <- theta
    }
    theta <- runif(1, theta_min, theta_max)
  }
}

```

```

update_z <- function(parameters, priors, data){

  # Choosing ellipse (nu) from prior (g)
  nu <- as.vector(MASS::mvrnorm(n = 1, mu = rep(0, length(parameters$z)), Sigma =
  ↪ parameters$sigma_2 * exp(-data$dists/parameters$phi)))

  # Log likelihood threshold (finding log(y))

  u <- runif(1, min = 0, max = 1)

  log_y <- loglike(parameters, data) + log(u)

  # Draw an initial proposal for theta

  theta <- runif(1, min = 0, max = 2*pi)
  theta_min <- theta - 2*pi
  theta_max <- theta

  repeat {
    # Calculate z'
    z_prime <- as.vector(parameters$z*cos(theta) + nu*sin(theta))

```

```

params_prime <- parameters
params_prime$z <- z_prime

# Shrinking bracket
if (loglike(params_prime, data) > log_y){
  parameters$z <- z_prime
  return(parameters)
} else {
  if(theta < 0) {
    theta_min <- theta
  } else {
    theta_max <- theta
  }
  theta <- runif(1, theta_min, theta_max)
}
}
}

# Updating LGCP Gaussian Latent field variance – sigma_2

update_sigma_2 <- function(parameters, priors, data){
  n <- length(parameters$z)

  alpha_post <- priors$a_0_sigma + n/2
  beta_post <- priors$b_0_sigma + 0.5 * t(parameters$z) %*% parameters$Sigma.Inv
  ↪ %*% parameters$z

  # Draw samples from Inverse–Gamma
  parameters$sigma_2 <- 1 / rgamma(1, shape = alpha_post, rate = beta_post)

  return(parameters)}

# Updating source 2 variance using Gibbs Sampling – tau_2

update_tau_2 <- function(parameters, priors, data){
  n <- length(parameters$g)

  alpha_post <- priors$a_0_tau + n/2
  beta_post <- priors$b_0_tau + 0.5 * t(parameters$g – parameters$alpha) %*%
  ↪ parameters$Sigma.Inv %*% (parameters$g – parameters$alpha)

  # Draw samples from Inverse–Gamma
  parameters$tau_2 <- 1 / rgamma(1, shape = alpha_post, rate = beta_post)

  return(parameters)
}

```

```

}

## Updating source 2 mean using Gibbs Sampling – alpha

update_alpha <- function(parameters, priors, data){
  n <- length(parameters$g)

  denom <- (sum(parameters$Sigma.Inv) / parameters$tau_2) + (1 / 100)
  mu_post <- (( sum(parameters$Sigma.Inv %*% parameters$g) / parameters$tau_2 ))
  ↪ / denom
  sigma_post <- sqrt(1 / denom)

  parameters$alpha <- rnorm(1, mean = mu_post, sd = sigma_post)

  return(parameters)
}

## Wrapper function

driver <- function(parameters, priors, data, iters ){
  out=list ()
  out$params=parameters
  out$priors=priors
  out$iters=iters

  #Posterior containers
  out$beta=matrix(NA,nrow = length(parameters$beta),ncol = iters)
  out$g=matrix(NA, nrow = length(parameters$g), ncol = iters)
  out$z=matrix(NA, nrow = length(parameters$z), ncol = iters)
  out$sigma_2=matrix(NA, nrow = 1, ncol = iters)
  out$tau_2=matrix(NA, nrow = 1, ncol = iters)
  out$alpha=matrix(NA, nrow = 1, ncol = iters)

  for(k in 1:iters ){
    parameters <- update_betas(parameters, priors, data)
    out$beta[,k]=parameters$beta

    parameters <- update_g(parameters, priors, data)
    out$g[,k] <- parameters$g

    parameters <- update_z(parameters, priors, data)
    out$z[,k] <- parameters$z

    parameters <- update_sigma_2(parameters, priors, data)

```

```

out$sigma_2[,k] <- parameters$sigma_2

parameters <- update_alpha(parameters, priors, data)
out$alpha[,k] <- parameters$alpha

parameters <- update_tau_2(parameters, priors, data)
out$tau_2[,k] <- parameters$tau_2
}
return(out)
}

# Running Simulation -----

data <- list(X_grid = X_grid,
            X_1 = X_1,
            X_2 = X_2,
            grid_res = 10,
            cell_area = (diff(win$xrange) / grid_res) * (diff(win$yrange) / grid_res
↪ ),
            nn_index_1 = nn_X_1$which,
            nn_index_2 = nn_X_2$which,
            win = win,
            cell_size = cell_size ,
            x_seq = x_seq,
            y_seq = y_seq,
            coords = as.matrix(expand.grid(y_seq, x_seq)),
            dists = as.matrix(dist(coords)))

parameters <- list(beta = c(0,0) ,
                  g = g,
                  z = z,
                  sigma_2 = sigma_2,
                  alpha = alpha,
                  tau_2 = tau_2,
                  phi = 4)

parameters$Sigma <- exp(-data$dists/ parameters$phi)
parameters$Sigma.Inv <- solve(parameters$Sigma)

priors <- list(beta_mean = c(0,0),
              beta_sd = c(10,10),
              beta_prop_sd = c(0.075, 0.075),
              z_mean = 0,
              a_0_sigma = 3,
              b_0_sigma = 1,

```

```

        a_0_tau = 2,
        b_0_tau = 1
    )

iters <- 10000

burnin <- 3000

sim <- driver(parameters, priors, data, iters )

# Running multiple simulations to calculate credible intervals -----

n_sims <- 100
n_sims_df <- data.frame()

S <- 0.2 * exp(-dists/4)

mu <- rep(0, n)

covariate <- as.vector(rcpp_rmvnorm(1,S,mu))

for (i in 1:n_sims){
  cat("Running simulation", i, "of", n_sims, "\n")

  win <- owin(xrange = c(0, 10), yrange = c(0, 10))

  grid_res <- 10

  cell_size <- diff(win$xrange) / grid_res

  x_seq <- seq(win$xrange[1] + cell_size/2,
              win$xrange[2] - cell_size/2,
              by = cell_size )
  y_seq <- seq(win$yrange[1] + cell_size/2,
              win$yrange[2] - cell_size/2,
              by = cell_size )

  coords <- as.matrix(expand.grid(y_seq, x_seq))
  grid_coords <- expand.grid(x = y_seq, y = x_seq)

  dists <- as.matrix(dist(coords))
  n <- nrow(coords)

  cov_field <- matrix(covariate - mean(covariate),
                    nrow = grid_res,

```

```

        ncol = grid_res,
        byrow = TRUE)

cov_field_ppp <- ppp(x = grid_coords$x,
                    y = grid_coords$y,
                    window = win,
                    marks = covariate)

#Simulate Gaussian random field

sigma_2 <- 0.10

S_z <- sigma_2 * exp(-dists/4)

z <- as.vector(rcpp_rmvnorm(1,S_z,mu))

z_mat <- matrix(z, nrow = length(x_seq), ncol = length(y_seq))

z_ppp <- ppp(x = grid_coords$x,
            y = grid_coords$y,
            window = win,
            marks = z)

# Simulate LGCP
b_0 <- 1.75
b_1 <- 3

lambda <- exp(b_0 + b_1*(cov_field) + z)
lambda_im <- im(lambda, xcol = x_seq, yrow = y_seq)

lgcp_sim <- rpoispp(lambda_im)
lgcp_sim

plot(lgcp_sim, main = paste("lgcp_sim", i))

# Discretize using spatstat

lgcp_discretize <- pixellate(lgcp_sim, eps = 1)

# Source 1 -----
nrow <- length(lgcp_discretize$yrow)
ncol <- length(lgcp_discretize$xcol)

x_min_subwindow1 <- 0
x_max_subwindow1 <- 5

```

```

y_min_subwindow1 <- 5
y_max_subwindow1 <- 10

x_min_subwindow2 <- 5
x_max_subwindow2 <- 10
y_min_subwindow2 <- 0
y_max_subwindow2 <- 5

sub_window1 <- owin(xrange = c(x_min_subwindow1, x_max_subwindow1), yrange
  ↪ = c(y_min_subwindow1, y_max_subwindow1))

sub_window2 <- owin(xrange = c(x_min_subwindow2, x_max_subwindow2), yrange
  ↪ = c(y_min_subwindow2, y_max_subwindow2))

lambda_1 <- lgcp_discretize

lgcp_1_sub1 <- rpoispp(lambda_1[sub_window1])
lgcp_1_sub2 <- rpoispp(lambda_1[sub_window2])

lgcp_1 <- superimpose(lgcp_1_sub1, lgcp_1_sub2, W = owin(c(0,10), c(0,10)))

# Source 2 -----

tau_2 <- 0.4
S_g <- tau_2 * exp(-dists/4)
alpha <- -0.2

g <- as.vector(rcpp_rmvnorm(1,S_g,alpha))

exp_g <- exp(g)

lambda_2 <- lgcp_discretize * exp_g

lgcp_2 <- rpoispp(lambda_2)

# Data frame creation -----

## X grid
X_grid <- as.data.frame(coords)
colnames(X_grid) <- c("x", "y")
X_grid$covariate <- covariate

## Source 1 (small var)
X_1 <- as.data.frame(lgcp_1)
nn_X_1 <- nncross(lgcp_1, cov_field_ppp)

```

```

X_1$covariate <- cov_field_ppp$marks[nn_X_1$which]

# making a mask for X-grid to be used with source 1
inside_sub1 <- with(X_grid,
  x >= x_min_subwindow1 & x <= x_max_subwindow1 &
  y >= y_min_subwindow1 & y <= y_max_subwindow1)

inside_sub2 <- with(X_grid,
  x >= x_min_subwindow2 & x <= x_max_subwindow2 &
  y >= y_min_subwindow2 & y <= y_max_subwindow2)

X_grid$mask_source1 <- inside_sub1 | inside_sub2

## Source 2 (large var)
X_2 <- as.data.frame(lgcp_2)
nn_X_2 <- nncross(lgcp_2, cov_field_ppp)
X_2$covariate <- cov_field_ppp$marks[nn_X_2$which]

data <- list(X_grid = X_grid,
  X_1 = X_1,
  X_2 = X_2,
  grid_res = 10,
  cell_area = (diff(win$xrange) / grid_res) * (diff(win$yrange) /
↪ grid_res),
  nn_index_1 = nn_X_1$which,
  nn_index_2 = nn_X_2$which,
  win = win,
  cell_size = cell_size ,
  x_seq = x_seq,
  y_seq = y_seq,
  coords = as.matrix(expand.grid(y_seq, x_seq)),
  dists = as.matrix(dist(coords)))

parameters <- list(beta = c(0,0) ,
  g = g,
  z = z,
  sigma_2 = sigma_2,
  alpha = alpha,
  tau_2 = tau_2,
  phi = 4)

parameters$Sigma <- exp(-data$dists/ parameters$phi)
parameters$Sigma.Inv <- solve(parameters$Sigma)

priors <- list(beta_mean = c(0,0),

```

```

        beta_sd = c(10,10),
        beta_prop_sd = c(0.075, 0.075),
        z_mean = 0,
        a_0_sigma = 3,
        b_0_sigma = 1,
        a_0_tau = 2,
        b_0_tau = 1
    )

    iters <- 10000

    burnin <- 3000

    # Run simulation -----

    sim <- driver(parameters, priors, data, iters )

    beta_post <- sim$beta[, (burnin+1):iters]
    alpha_post <- sim$alpha[, (burnin+1):iters]
    sigma_2_post <- sim$sigma_2[, (burnin+1):iters]
    tau_2_post <- sim$tau_2[, (burnin+1):iters]
    g_post <- sim$g[, (burnin+1):iters]
    z_post <- sim$z[, (burnin+1):iters]

    # Calculating posterior
    posterior_lambda <- matrix(NA, nrow = nrow(X_grid), ncol = (iters-burnin))

    for(m in 1:(iters-burnin)){
        beta_m <- beta_post[,m]

        log_lambda_m <- beta_m[1] + beta_m[2]*covariate + z_post[,m]

        posterior_lambda[, m] <- exp(log_lambda_m)
    }

    # Posterior mean intensity
    lambda_mean <- rowMeans(posterior_lambda, na.rm = TRUE)

    actual_count <- sum(lambda * data$cell_area, na.rm = TRUE)

    n_draws <- ncol(posterior_lambda)
    expected_total_per_draw_fused <- numeric(n_draws)

    for (m in 1:n_draws) {
        expected_total_per_draw_fused[m] <- sum(posterior_lambda[, m] * data$cell_area

```

```

  ↪ , na.rm = TRUE)
}

expected_mean_fused <- mean(expected_total_per_draw_fused)
expected_sd_fused <- sd(expected_total_per_draw_fused)

# Store just summary stats
n_sims_df <- rbind(n_sims_df, data.frame(
  sim = i,
  sim_n = lgcp_sim$n,

  # beta parameters
  beta0_mean = mean(beta_post[1,]),
  beta0_sd = sd(beta_post[1,]),
  beta0_lower = quantile(beta_post[1,], 0.025),
  beta0_upper = quantile(beta_post[1,], 0.975),

  beta1_mean = mean(beta_post[2,]),
  beta1_sd = sd(beta_post[2,]),
  beta1_lower = quantile(beta_post[2,], 0.025),
  beta1_upper = quantile(beta_post[2,], 0.975),

  # hyperparameters
  sigma2_mean = mean(sigma_2_post),
  sigma2_sd = sd(sigma_2_post),
  sigma2_lower = quantile(sigma_2_post, 0.025),
  sigma2_upper = quantile(sigma_2_post, 0.975),

  tau2_mean = mean(tau_2_post),
  tau2_sd = sd(tau_2_post),
  tau2_lower = quantile(tau_2_post, 0.025),
  tau2_upper = quantile(tau_2_post, 0.975),

  alpha_mean = mean(alpha_post),
  alpha_sd = sd(alpha_post),
  alpha_lower = quantile(alpha_post, 0.025),
  alpha_upper = quantile(alpha_post, 0.975),

  g_mean = mean(g_post),
  g_sd = sd(g_post),
  g_lower = quantile(g_post, 0.025),
  g_upper = quantile(g_post, 0.975),

  z_mean = mean(z_post),
  z_sd = sd(z_post),

```

```
z_lower = quantile(z_post, 0.025),
z_upper = quantile(z_post, 0.975),

#expected counts
count_mean = expected_mean_fused,
count_lower = quantile(expected_total_per_draw_fused, 0.025),
count_upper = quantile(expected_total_per_draw_fused, 0.975),
actual_count = actual_count

))
}
```

References

- S. Banerjee, B.P. Carlin, and A.E. Gelfand. *Hierarchical Modeling and Analysis for Spatial Data*. Chapman and Hall/CRC, 2 edition, 2014. doi: <https://doi.org/10.1201/b17115>.
- J. Bernardo, J. Berger, A. Dawid, Katja Ickstadt, and Robert Wolpert. Spatial regression for marked point processes. 09 1998.
- Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013:704504, 2013. doi: 10.1155/2013/704504. URL <https://doi.org/10.1155/2013/704504>.
- Christopher W. Clark, Moira W. Brown, and Peter Corkeron. Visual and acoustic surveys for North Atlantic right whales, *Eubalaena glacialis*, in Cape Cod Bay, Massachusetts, 2001–2005: Management implications. *Marine Mammal Science*, 2010. doi: 10.1111/j.1748-7692.2010.00376.x.
- Noel A. C. Cressie. *Statistics for Spatial Data: Revised Edition*. John Wiley & Sons, New York, revised edition, 1993. ISBN 978-0-471-00255-0.
- B. V. Dasarathy. Sensor fusion potential exploitation: Innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997. doi: 10.1109/5.554206.
- Alan E. Gelfand and Erin M. Schliep. *Bayesian Inference and Computing for Spatial Point Patterns*. NSF-CBMS Regional Conference Series in Probability and Statistics. Institute of Mathematical Statistics, 2018. URL <https://www.jstor.org/stable/26477714>. CBMS Short Course, University of California at Santa Cruz.
- Alan E. Gelfand and Erin M. Schliep. Model-based spatial data fusion. *Annual Review of Statistics and Its Application*, 12, 2025. doi: 10.1146/annurev-statistics-042424-052920. URL <https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-042424-052920>.
- David A. Hofmann and Mark B. Gavin. Centering decisions in hierarchical linear models: Implications for research in organizations. *Journal of Management*, 24(5):623–641, 1998. ISSN 0149-2063. doi: [https://doi.org/10.1016/S0149-2063\(99\)80077-4](https://doi.org/10.1016/S0149-2063(99)80077-4). URL <https://www.sciencedirect.com/science/article/pii/S0149206399800774>.
- H. B. Mitchell. *Data Fusion: Concepts and Ideas*. Springer Berlin, Heidelberg, 2 edition, 2012.
- Iain Murray, Ryan P. Adams, and David J. C. MacKay. Elliptical slice sampling. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *Journal of Machine Learning Research: Workshop and Conference Proceedings*, pages 541–548. JMLR.org, 2010. URL <http://jmlr.org/proceedings/papers/v9/murray10a/murray10a.pdf>.
- Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998. doi: 10.1111/1467-9469.00115.

David Poole and Adrian E. Raftery. Inference for deterministic simulation models: The bayesian melding approach. *Journal of the American Statistical Association*, 95(452): 1244–1255, 2000. ISSN 01621459, 1537274X. URL <http://www.jstor.org/stable/2669764>.

Brian J. Reich and Sujit K. Ghosh. *Bayesian Statistical Methods*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2019.

Erin Schliep. ST 533: Applied spatial statistics. Course notes, 2025.

Erin M. Schliep, Alan E. Gelfand, Christopher W. Clark, Charles M. Mayo, Brigid McKenna, Susan E. Parks, Tina M. Yack, and Robert S. Schick. Assessing marine mammal abundance: A novel data fusion. *Annals of Applied Statistics*, 2024.

Patrick Ziegler and Klaus R. Dittrich. Data integration—problems, approaches, and perspectives. In John Krogstie, Andreas L. Opdahl, and Sjaak Brinkkemper, editors, *Conceptual Modelling in Information Systems Engineering*. Springer, 2007.

GitHub Repository for Simulation Data and Code:

<https://github.com/egshipp/Shipp-Thesis-Data-Fusion-for-Spatial-Point-Processes>