

Exploring Anisotropic Kernel Methods in Spatial Data Modeling: Review and Case Study

STATS 771 - Statistical Research Project Report

John Tweedie

Student number: 400550023

2024-04-22

Introduction

The philosophical basis of spatial analysis can be most-broadly summarized by the notion that points in space are more-related to points closer-in than points further in proximity (Tobler, 1970). This notion runs in contradiction to the often-held assumption of independence and identical distribution amongst residual values of predictive statistical and machine learning models (F. Dormann et al., 2007; Liu et al., 2022). Evidently, these two schools of thought need to be reconciled to achieve accurate and unbiased model-based predictions across relevant disciplines. With the advent of modern data capturing and processing methods, much data reaches the end-user in spatial (i.e. geographical) formats, often forming a continuous or pseudo-continuous representation of random variables across space (Goodchild, 2013). Typical examples of spatial data can be observed from satellite imagery, government census mapping, or computed tomography (CT) scans (Van Lieshout, 2019). Applications of spatial data, especially the statistical and machine-learning based approaches for analysis and prediction, have risen in parallel (Liu et al., 2022). As such, it is important that these applications employ appropriate methods to consider and account for the spatial dependence structures that exist in these data.

Implications of spatial dependencies in associated statistical analyses have been formally documented dating back to 1914 (Bataineh et al., 2006). Since objects in space tend to exhibit influence or similarity (of some degree) to neighboring objects, their locations contain value information that can be leveraged to enhance predictive capabilities. However, if left unchecked, this information can act as a hindrance to prediction/classification, unbiased parameter estimation, and hypothesis testing due to autocorrelation of residuals (F. Dormann et al., 2007; Fornango, 2010; Portier et al., 2018). Currently various methodological frameworks for sampling and modelling exist with the shared general intention of neutralizing residual spatial autocorrelation (Bataineh et al., 2006). Extensions of traditional regression methods are common practices to address SAC; examples include geographically weighted regression, which allows for spatially-varied regression coefficients (Kupfer and Farris, 2007), spatial eigenvector methods, which attempt to decompose spatial dependence structures and incorporate transformed covariates into generalized linear models (F. Dormann et al., 2007), spatial lag models that explicitly incorporate weighted information of neighboring values (Darmofal, 2015), and Bayesian spatial models that incorporate terms for spatially structured random effects (Banerjee, 2015; Chun et al., 2012). Detection and quantification of spatial autocorrelation is possible using indices and descriptive statistics, such as Moran’s I, Geary’s c , or Getis-Ord Local G_i (Banerjee, 2015; Darmofal, 2015). Modelling the dependence characteristics between observations as a function of separation distance can be completed by fitting an appropriate theoretical variogram models (Legendre, 1993).

Predictive modelling applications in spatial contexts are often related to meteorological, environmental, ecological, epidemiological, and image/object pattern-detection analyses (Kanjir et al., 2018; Muhammad et al., 2021; Van Lieshout, 2019). Modern applications benefit from the advent of efficient and accessible spatial data collection and processing methods, allowing practitioners to predict phenomena continuously at resolutions and scales not previously feasible (Liu et al., 2022; Muhammad et al., 2021). Two applications that have particularly benefited are post-fire forest burn area classification and container ship vessel detection from satellite imagery (Hawbaker et al., 2020; Kanjir et al., 2018). Robust literature and many well-refined predictive modelling and classification frameworks (e.g. machine-learning algorithms) exist to address these topics; however,

these predictions remain challenging and/or computationally demanding (Muhammad et al., 2021). Consideration of spatial autocorrelation is often overlooked and excluded from such predictive models, which often aim to classify individual points or pixels assumed as independent (Fornango, 2010; Portier et al., 2018). Utilizing the spatial autocorrelation information of residuals may therefore offer an avenue for improvement in terms of prediction accuracy for such applications, as spatial dependence between relevant predictors is common (Kupfer and Farris, 2007; Liu et al., 2022; Ploton et al., 2020; Portier et al., 2018). For instance, Portier et al., 2018 show an improvement in forest burn rate model performance by a factor of 1.5 when employing a spatial component by way of a residual autocovariate model, and conversely, Ploton et al., 2020 illustrate how exclusion of spatial autocorrelation measures can lead to false and highly misleading interpretations of standard model predictive power.

This study will aim to provide insight regarding current methods to account for spatial dependencies in the context of predictive modelling and classification using geo-spatial satellite image data, and how these methods perform in common applications (i.e. burn area classification and vessel detection). A review of pertinent literature and text is provided to document a full step-wise procedure for modelling spatial data, and a case study application is provided thereafter to illustrate the process.

Background Review

Spatial Processes

We can formally define a multivariate spatial process by a random vector $Y(s)$, where $s \in \mathbb{R}^d$ is a location in d -dimensional space. Spatial processes are typically observed throughout a subset of $\mathbb{R}^d$, denoted as index set D . Banerjee, 2015 and Darmofal, 2015 define three basic types of spatial data; *point-referenced*, *areal*, and *point pattern* data. In point-referenced data, the location s is defined continuously throughout D . For instance, observations associated with latitude and longitude coordinates in census maps represent point-referenced data. In areal data, D is partitioned into a finite amount of discrete units in either a regular or an irregular manner (Darmofal, 2015). A common example of regular areal data is in image data, where the location index pertains to the row and column of a pixel. Common examples of areal data with irregular discretization includes census tract, county, and legislative district data (Darmofal, 2015). Lastly, in point pattern data, the index subset D is, itself, a random spatial process, where $Y(s) \in [0, 1]$ defines the occurrence (1) or absence (0) of the spatial process at location s . Typical examples of point pattern data arise in ecological contexts; particularly in species-distribution mapping. Spatial clustering characteristics (or lack thereof), such as local intensity, are often the main point of interest when analyzing point pattern data. For the purpose of this study, we restrict the definition of spatial processes to point-referenced and areal stochastic processes occurring within a subset D two-dimensional euclidean space ($\mathbb{R}^2$).

Spatial Autocorrelation

The presence of spatial autocorrelation is a hallmark property of spatial data. In a general sense, it can be described as the tendency for pairs of points to exhibit more-similar or less-similar values at certain separation distances than would be expected from a truly random process (i.e. devoid of any spatial autocorrelation) (Legendre, 1993). Spatial autocorrelation can arise either endogenously from the process itself, such as vegetation

growth distributions, or can be induced exogenously, such as climate forces influencing patterns in landforms (Fortin and Dale, 2005). Spatial autocorrelation generally exhibits three characteristic properties; scale-dependence, structure, and directionality (Yu, 2010). Scale-dependence refers to how spatial autocorrelation patterns may change, be present/not-present, or be the dominant/non-dominant process when viewed at different scales or resolutions. For instance, small-scale landscape features (i.e. ditches, mounds) can often not be represented in lower-resolution topographic spatial data (Legendre, 1993). The structure of spatial autocorrelation refers to whether processes exhibit a pattern defined by a gradient trend surface, discrete patches or 'clumps', or randomness. Lastly, directionality reflects whether spatial autocorrelation patterns exhibit non-isotropic patterns; i.e. spatial autocorrelation that is stronger in certain directions and weaker in others. It is important to consider these three main characteristics in the context of the spatial process at hand in order to appropriately characterize, quantify, and account for spatial autocorrelation.

In model-based statistical analyses, accounting for spatial autocorrelation is often completed by first constructing a spatial weight matrix to describe the underlying dependencies between observations (Legendre, 1993). An appropriately specified spatial weight matrix should capture and reflect the underlying characteristics of the stochastic spatial process in terms of scale, structure, and directionality (Yu, 2010). Many spatial weight specification models exist; however, literature describing generalized guidelines for model specification remains relatively sparse (Lam and Souza, 2020; Stakhovych and Bijmolt, 2009). We define an $n \times n$ spatial weights matrix W as a non-negative matrix with elements w_{ij} that specify the set of neighboring observation. Typically, self-neighboring observations are excluded (i.e. $w_{ii} = 0$), and matrices are utilized in row-standardized form. Stakhovych and Bijmolt, 2009 identify three distinct classes of spatial weight matrix specification approaches; *distance/neighborhood-related*, *model-based*, and *data-driven*. Distance- and neighborhood-related methods treat W as exogenous to the stochastic spatial process, and define weighting schemes based on specified functions of distance between observations (e.g. inverse distance weighting) or functions of partitioned spatial units containing neighboring observations (e.g. N nearest neighbors). Model-based methods generally refer to a approaches that estimate W by specifying models endogenous to the underlying data, or latent variables. Similarly, approaches that allow properties of the underlying data itself (e.g. the presence of distinct clusters) to construct W are termed as *data-driven*.

Specification of an appropriate spatial weights matrix is vital for accurate and unbiased parameter estimation during later stages of spatial analyses (Lam and Souza, 2020; Stakhovych and Bijmolt, 2009). For various practical reasons, constructing W to best represent spatial dependence structures requires various assumptions and constraints regarding the characteristics of the spatial process. A fundamental concept central to these assumptions and constraints is the notion of *stationary* spatial processes. Various degrees of stationarity exist, and their associated assumptions and conditions are necessary to achieve appropriate application and replication of functions and models throughout the full extent of the spatial process (Banerjee, 2015; F. Dormann et al., 2007). Assuming there exists a mean $\mu(s) = E(Y(s))$, a spatial process $Y(s)$ is considered *strictly stationary* if for any $n \geq 1$ set of observations $s_1, \dots, s_n$ and any lag distance $h \in \mathbb{R}^d$, the distribution of $Y(s_1), \dots, Y(s_n)$ will be equivalent to that of $Y(s_1 + h), \dots, Y(s_n + h)$ (Banerjee, 2015). Similarly, a spatial process is termed *weakly stationary* if the mean is constant through space and the covariance is dependent on the spatial lag h ; i.e. $\mu(s) = \mu$

and $Cov(Y(s), Y(s+h)) = C(h) \forall h \in \mathbb{R}^d$, where $C(h)$ is a specified covariance function (Banerjee, 2015).

For spatial processes, there exists a third notion of stationarity by way of *intrinsic stationarity* (Banerjee, 2015). If the variance is only dependent on lag h , and not on any particular location s , then the process is considered intrinsically stationary. More formally, an intrinsically stationary process will exhibit the following:

$$E[Y(s+h) - Y(s)]^2 = Var(Y(s+h) - Y(s)) = 2\gamma(h) \quad (1)$$

where $E[Y(s+h) - Y(s)] = 0$. The function $2\gamma(h)$ is called the *variogram*; an important function that describes the degree of spatial dependence between two points residing $\mathbf{h}$ units of distance apart. When the process is intrinsically stationary (or assumed to be), the variogram is only dependent on lag distance h and is independent of location s . A *semi-variogram* $\gamma(h)$ is simply one-half of the variogram. While variograms describe the behaviour of differences between observations, they do not provide information regarding the joint distribution of observations and their associated likelihood (Banerjee, 2015). Regardless, variograms (and semi-variograms) are an essential component to modelling spatial processes and constructing spatial weighting matrices. Often, finding a variogram model specification that appropriately reflects the observed spatial dependence structure can serve as the mathematical basis for assigning lag distance-based weights w_{ij} in W .

Variogram Specification

A variety of variogram models exist, which generally differ by distance and directional spatial weighting characteristics. In practice, variogram models should be specified in consideration of the underlying spatial dependence properties of the process (Legendre, 1993; Yu, 2010). Denoting $d = \|\mathbf{h}\|$, a simple example is that of a *linear semi-variogram* model, defined as follows:

$$\gamma(d) = \begin{cases} \tau^2 + \sigma^2 d & d > 0 \\ 0 & o.w. \end{cases} \quad (2)$$

where τ^2 is the *nugget effect* and σ^2 is the *sill*. The nugget effect represents the value of the variogram at distance $d = 0$, and the sill represents the asymptotic behaviour of the variogram (i.e. values at lags approaching infinity). Additional options describing a general decay in weightings include spherical, Gaussian, exponential, and rational quadratic, amongst others (Banerjee, 2015; Getis and Aldstadt, 2004). Further, options for periodic weightings exist, which generally use combinations of sine and cosines to model the perceived spatial trend (Legendre, 1993). A spherical semi-variogram is defined as (Getis and Aldstadt, 2004):

$$\gamma(d) = \begin{cases} \sigma^2 \left(\frac{3d}{2r} - \frac{d^3}{2r^3} \right) & d < r \\ \sigma^2 & o.w. \end{cases} \quad (3)$$

where r is the *range* parameter, representing the distance at which the variogram $\gamma(d)$ is equal to the sill σ^2 . Spherical variograms are valid for one, two, and three-dimensional feature spaces, but generate spatial matrices that are not positive definite for dimensionality greater than or equal to four (Banerjee, 2015). Gaussian semi-variograms are another common variogram model, which provide a 'smoothed' representation of the

spatial process. A Gaussian semi-variogram is defined by:

$$\gamma(d) = \sigma^2 (1 - \exp(-3d^2/r^2)) \quad d < r \quad (4)$$

Variogram models (and associated semi-variogram models) may be *isotropic* or *anisotropic* with respect to kernel geometry. Isotropic variograms are typically simpler in regards to their associated parameterizations, whereas anisotropic semi-variograms introduce an added layer of complexity, as the associated spatial weightings are no longer applied in an equi-directional manner. As such, the spatial dependence between points is no longer only dependent on the corresponding distance between locations, as it is also a function of direction; hence, the variogram can no longer be specified as $\gamma(d)$ (Banerjee, 2015). For instance (assuming a uniform variogram model that applies equal weights to all observations within a spatial unit) one can construct an anisotropic semi-variogram that takes an elliptical kernel shape oriented in a certain direction. Such approaches may allow for more-effective capture of spatial phenomena that display a tendency to align in parallel manners or with an underlying gradient (He et al., 2015). Literature pertaining to such approaches remains sparse, with the exception of He et al., 2015, who apply anisotropically foveated spatial weighting schemes in multi-spectral satellite image classification, and subsequently illustrate a boost in classification model performance.

Fitting an appropriate variogram model to the data is often completed by constructing the *empirical semivariogram* $\hat{\gamma}(d)$; a method of moments estimator for the sample variance estimate. The empirical semivariogram is defined as:

$$\hat{\gamma}(d) = \frac{1}{2N(d)} \sum_{N(d)} [Y(s_i) - Y(s_j)]^2 \quad (5)$$

where $N(d)$ is the set of all pairs of points that are euclidean distance $d = s_i - s_j$ apart, and N represents the number of pairs within the set (Getis and Aldstadt, 2004). Plotting the empirical semivariogram of the data can allow the practitioner to appropriately fit a theoretical variogram model, and thus, specify an appropriate spatial weighting matrix. Parameterization (i.e. sill and range) for a prescribed variogram model can typically be estimated by analyzing characteristics (e.g. asymptotic behaviour, y-intercept) of the empirical variogram plot (Banerjee, 2015; Getis and Aldstadt, 2004).

Spatial Modelling

Upon detecting and characterizing spatial autocorrelation of a dataset, one can apply a variety of models that utilize this information to enhance estimation accuracy relative to their non-spatial counterparts. The fundamental basis of these models is that they directly consider and account for the influence of neighboring observations, in addition to the central observation itself. Generally, there exists two broad methodological classes of spatial modelling approaches; *spatial lag* and *spatial filtering* (Liu et al., 2022). Spatial lag models (a.k.a. mixed regressive, spatial autoregressive models) are perhaps the most common and widely adopted model across relevant fields of application. Due to the multi-dimensional nature of spatial processes (i.e. autocorrelation is typically present in a diffuse manner, rather than unidirectional) ordinary least squares methods cannot be used for spatial lag model parameter estimation; rather, these models typically rely on maximum likelihood approaches (Darmofal, 2015). Spatial lag models generally take the

following parameterization:

$$Y = \rho WY + X\beta + \epsilon \quad (6)$$

where y is an $N \times 1$ vector containing the response variable, W is the constructed $N \times N$ spatial weighting matrix, ρ is the autocorrelation coefficient describing the dependence structure of the response, X is an $N \times p$ matrix of covariates, and β is the associated $p \times 1$ vector of regression parameter estimates (Darmofal, 2015). Fixed effects can be modelled by the inclusion of an additive parameter μ , representing an $N \times 1$ vector of constants (Lam and Souza, 2020). Here, the autocorrelation coefficient ρ adjusts the magnitude of spatial dependencies between observations (prescribed by W) used in fitting the model.

Various extensions and modifications of the basic spatial lag model exists. For instance, the spatial weight matrix can be decomposed into a sparse component A^* and 'known' component W_0 weighted by ρ , defined by $W^* = A^* + \rho^*W_0$. The sparse component represents the spatial autocorrelation unaccounted for by W_0 , and will therefore be zero when W_0 perfectly reflects the underlying spatial dependence structure. Decomposing W accordingly can allow one to perform a penalized estimation of A^* under a LASSO framework; effectively shrinking and zeroing elements of A^* to find an optimal specification of W^* (Lam and Souza, 2020). Additional extensions include spatial generalised linear models, spatial lag mixed models, Bayesian spatial models, and spatial error models, amongst others (Chun et al., 2012; F. Dormann et al., 2007).

An alternative approach to explicitly including the weighting matrix by the spatial lag term is to perform eigenvector spatial filtering of the spatial data (a.k.a. eigenvector mapping). This approach involves decomposing the spatial weights matrix W into its eigenvectors and eigenvalues. A subset of the 'best' resulting eigenvectors are selected, wherein those that reduce spatial autocorrelation are given preference. This subset comprises an additional matrix of covariates that explain the spatial dependence structure, and is typically added to the original covariate set (F. Dormann et al., 2007). This 'combined' data can then fit with generalized linear models, analogous to fitting with non-spatial data.

Evaluating the effectiveness of a proposed variogram, spatial weighting matrix, and spatial model can be done by calculating and investigating the Moran's I statistic; an index of spatial autocorrelation that can be applied locally and globally. For descriptions of global spatial autocorrelation, Moran's I is calculated as follows:

$$I = \frac{N \times \sum_{i=1}^N \sum_{j=1}^N w_{ij} (Y_i - \bar{Y}) (Y_j - \bar{Y})}{\sum_{i=1}^N \sum_{j=1}^N w_{ij} \times \sum_{i=1}^N (Y_i - \bar{Y})^2} \quad (7)$$

where N represents the number of observations. Moran's I is a useful statistic to test whether spatial autocorrelation exists within a given dataset; a calculated value of I that is significantly different than the associated I value expected under the null hypothesis of no spatial autocorrelation is indicative of the presence of positive or negative spatial autocorrelation. Under the null hypothesis, the expected value of I is determined as $\frac{-1}{N-1}$. I values range from -1 to 1, where positive values represent positive spatial correlation globally. If there remains a significant portion of spatial autocorrelation amongst associated residual values, then there is likely a alternative variogram model available to better capture and utilize this information. More specifically,

An alternative, yet similar statistic indicative of spatial autocorrelation is Geary's c ,

defined as:

$$C = \frac{(N - 1) \times \sum_{i=1}^N \sum_{j=1}^N w_{ij} (Y_i - Y_j)^2}{2 \left(\sum_{i=1}^N \sum_{j=1}^N w_{ij} \right) \times (Y_i - \bar{Y})^2}, \quad (8)$$

Interpretation of both statistics is largely analogous, wherein higher values are indicative of greater levels of spatial dependency among observations. Geary’s c is more-sensitive to extreme values due to the squared term in the numerator, and as such, Moran’s I is more commonly adopted in practice (Darmofal, 2015).

Case Study

A practical case study application was completed to illustrate the predictive value of the underlying spatial dependence structure in common image-based spatial data. The general framework was to compare predictions resulting from a baseline, non-spatial model to that of various models which incorporate spatial dependence components. More specifically, the baseline model was compared to an eigenvector spatial filter model and several spatial lag models. Various spatial weighting matrix construction techniques were employed for the spatial lag models, including varying neighborhood distance cutoffs and varying orientations of anisotropic neighborhoods. This serves to address the aforementioned gap in literature, wherein documented testing of construction methods and certain specifications of spatial weight matrices remains sparse (Lam and Souza, 2020; Stakhovych and Bijmolt, 2009). This appears especially true regarding available information describing anisotropic kernel methods and their implications.

Models were fit to three non-simulated datasets representing practical applications that would presumably benefit from the inclusion of spatial autocorrelation measures. Each dataset was a high-resolution multi-band satellite image, with two images depicting a forested area that had recently experienced a large-scale fire, and one of a container ship in open water that was partially obscured by cloud cover. Burn area prediction and container ship identification are two common tasks that utilize similar spatial data, and therefore, the case study could presumably provide direct practical insight regarding incorporation of spatial autocorrelation information (Kanjir et al., 2018; Muhammad et al., 2021; Pham et al., 2020).

All simulations were completed using R and various associated packages. A copy of the R scripts are provided in **Appendix A**, for reference.

Data Procurement and Processing

Burn Area Data

Burn area data for the simulation study was obtained from multiple sources. Multi-spectral satellite imagery data from the Landsat satellites was obtained from United States Geological Service (USGS) and served as the predictor variables for the burned-area modelling - a common application of the multi-spectral Landsat imagery (Muhammad et al., 2021; US Geological Survey, 2023). The pre-classified burned-area response data was obtained from Alberta Geospatial Services (Alberta Geospatial Services, 2023). For the purpose of this analysis, it was assumed that this burn area classification was ground-truthed and fully accurate. Burn area data processing was completed within

QGIS software (QGIS, 2023), and included aligning all imagery with burn-area delineation, resampling data at a 300 m pixel resolution, and re-exporting in a tabular format for use in R. Of the spectral bands provided in the LandSat image, only the 'band 6', 'trad', 'emis', 'drad', and 'emsd' spectral bands were retained for modelling, thus representing the predictor covariate set. The burned-area data represents the binary response variable to predict.

Two 'subsets' of the burn area data were extracted from the full satellite image, as applying the proposed modelling methods to such a large spatial dataset is not feasible due to computational and time constraints. Further, this allowed for two separate models to be fit to partitions of the same parent data, for which comparisons of resulting coefficient estimates and prediction accuracy can be made to illustrate the sensitivity of the methods to the data. The processed burn area multi-band satellite imagery data subsets *Subset 1* and *Subset 2* are shown in Figures 1 and 2, respectively. Subset 1 is characterized by cluster-like patterns of spatial autocorrelation, with narrowing interconnecting 'corridors' of burn area that generally follow a Northwest to Southeast orientation. Subset 2 is largely characterized by a single discrete boundary separating the burn area classes.

Container Ship Data

Container ship imagery data was obtained from Kaggle (Hammell, 2018). Various container ship satellite images are provided in a non-georeferenced, 80x80 pixel image format. One image was selected for analysis, based on uniformity of the ship's colours and the presence of clouds or ripples that could presumably impact detection methods. An RGB-channel noise filter was applied to the image to further degrade and disrupt the evident spatial autocorrelation of the ship pixels - this helped make model-classifications more reliant on the spatial autocorrelation components for accurate predictions. The raw and noise-added image is provided in Figure (3). To extract usable predictor covariates, the image was split into the associated red, green, and blue colour channels. Lastly, a mask layer was overlain manually to delineate the binary spatial response data (i.e. the pixels where the container ship is present vs. absent). The response variable and three covariates are shown in Figure 4, for reference.

Case Study Methods

The case study simulations included four general stages of modelling, followed by a comparative analysis of the results. Comparison of final model results was completed on the basis of prediction accuracy, f1-score, coefficient estimates, Moran's I of residuals, and by visual inspection of model predictions. Prior to fitting any models, empirical semivariograms were generated and global Moran's I was calculated to inform potential model parameterizations for spatial weight construction. Spatial plots of the raw data (i.e. Figures 1, 2, and 4) were also investigated in this regard.

Given the binary responses in both datasets, all models were fit using a logistic generalized linear model (GLM) framework. Initially, a 'baseline' logistic GLM was fit using the *glm()* function with a logit link. The results of this non-spatial baseline model served as a benchmark for which to compare the subsequent spatial model results. The remaining three modelling exercises utilized various methods of accounting-for and characterizing spatial autocorrelation.

Two spatial modelling exercises used a spatial lag model to account for the spatial

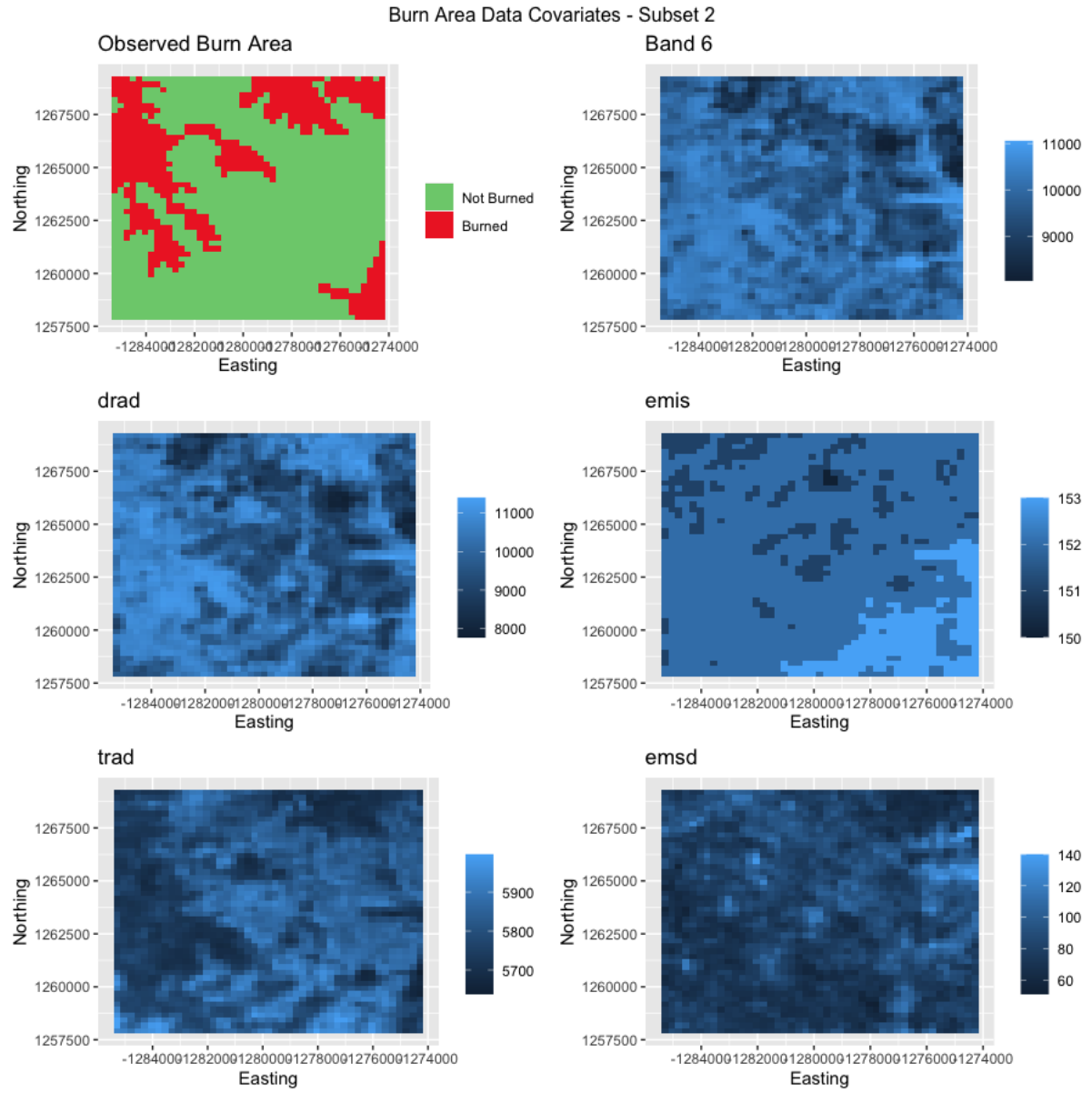


Figure 1: Burn area satellite imagery data partitioned subset 1

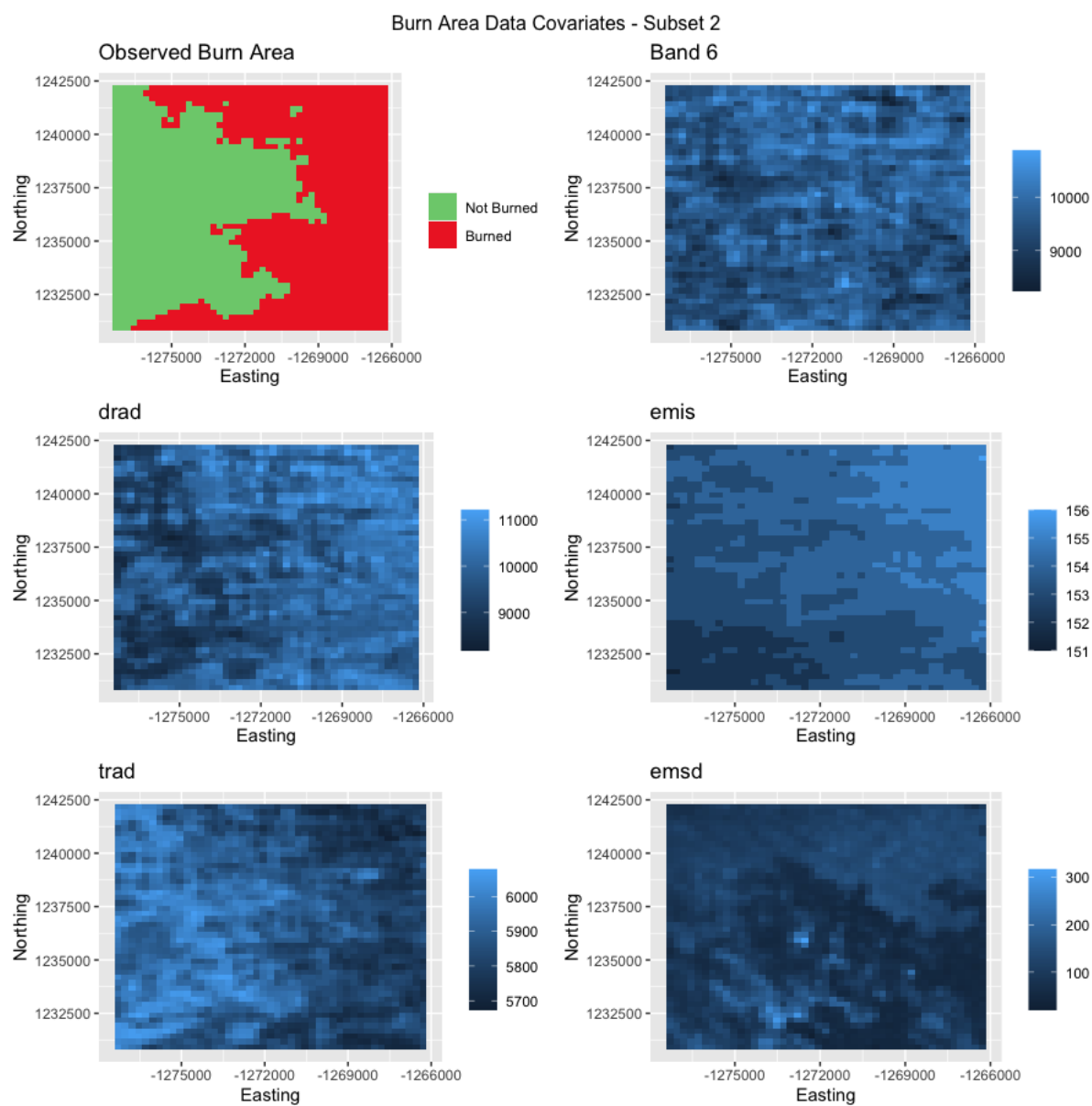


Figure 2: Burn area satellite imagery data partitioned subset 2



(a) Raw container ship image data



(b) Noise-added container ship image data

Figure 3: Raw and noise-added container ship image data

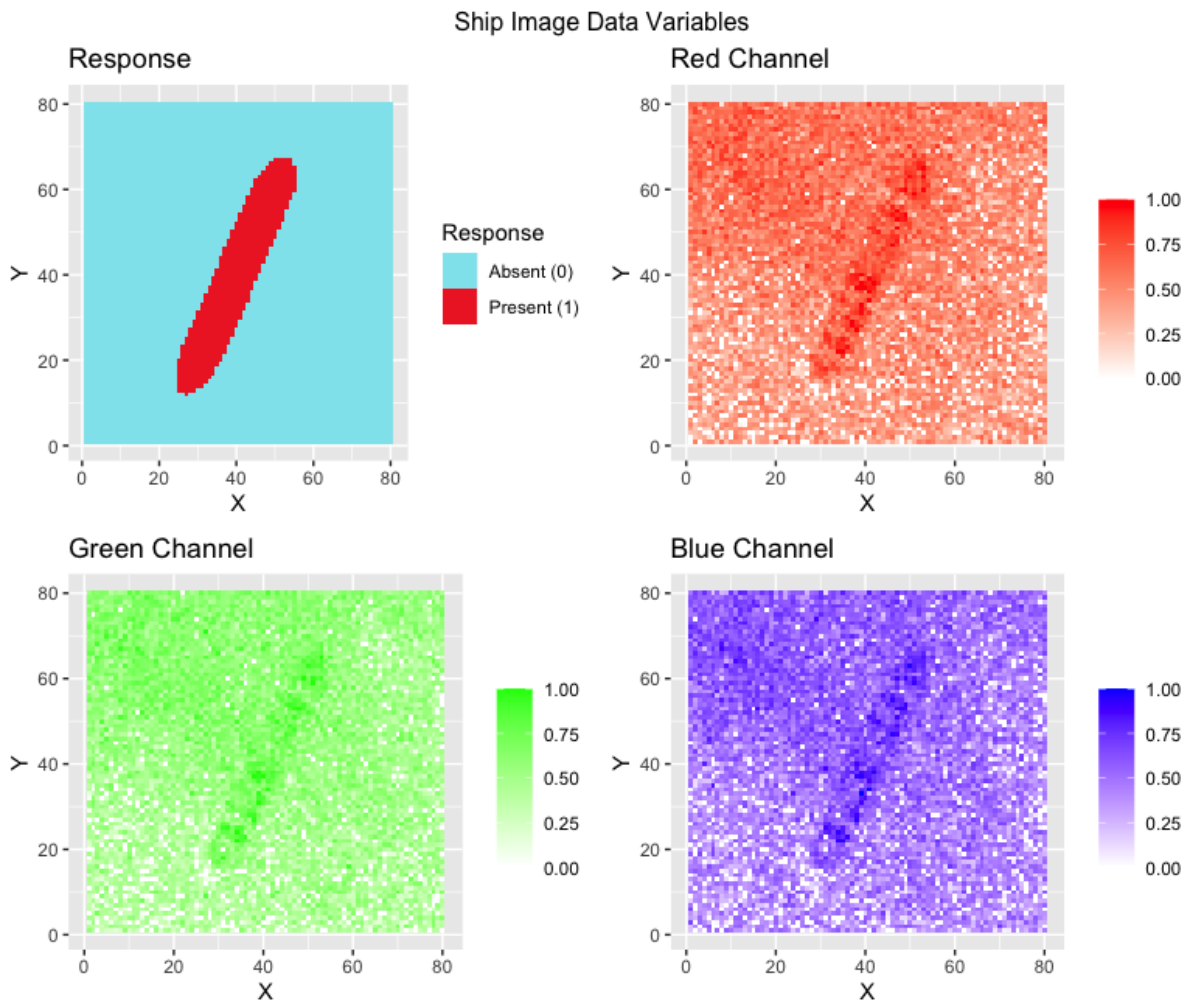


Figure 4: Container ship data covariate plots

autocorrelation, but differed in their approaches to construction of the spatial weight matrix W . The first spatial lag modelling method constructed W using a distance-based nearest neighborhood. Uniform weightings were assigned to all neighboring observations falling within a specified cutoff distance. In essence, this represents a variogram model simply defined by a local average. Various cutoff distances were employed to generate multiple specifications of W and associated model fits. Spatial weight matrices were specified using the *dnearneigh()* function, and spatial lag models were fit using the *lagsarlm()* function, both from the *spdep* package. It is noted that *lagsarlm()* does not offer a generalized linear logistic model functionality. As such, a pseudo-GLM approach was taken where the *lagsarlm()* was treated as the linear predictor, and fitted values (constrained to a [0,1] interval and estimated via maximum likelihood) were subsequently transformed using an inverse logit link function. While not an ideal method, this was considered adequate given the focus of the analysis was on the spatial autocorrelation components, and the resulting fitted values appeared very reasonable.

The second spatial lag model method utilized spatial weight matrices derived in a similar manner (d-nearest neighbor with uniform weighting), but instead, using an anisotropic (elliptical) neighborhood kernel. Construction of W using an anisotropic kernel was completed by scaling transforming the underlying data itself, then fitting the *dnearneigh* model as before. The scaling and transformation of the data was done so in a manner inverse to that of directly specifying an elliptical neighborhood kernel, as this is not an option within *dnearneigh*. For instance, an elliptical kernel with twice the width (parallel to the x-axis) relative to the height (y-axis) can be achieved by scaling the data such that the x-coordinates (and subsequently, the horizontal distance between observations) are halved. Thus, the circular kernel within *dnearneigh* will capture twice the observations radiating horizontally from the centroid observation relative to those radiating vertically, subsequently producing a neighborhood of points exactly analogous to that which would be expected from an elliptical kernel. In a similar manner, observations are rotated and then scaled to achieve directional tendencies unconstrained to the x- and y-axes. More formally, the data transformations were completed as follows; given a vector S containing x and y observation coordinates, rotation and is completed by applying the rotation matrix R defined by:

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (9)$$

where θ is the desired angle (radians) of rotation. As such, the rotated version of our coordinate vector S' is determined by:

$$S' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (10)$$

The resulting X and Y coordinate vectors are then multiplied by a specified scalar value to stretch/compress the spatial distribution, as required. In this instance, a scaling factor of 0.2 was applied to the X coordinate vector; this corresponds to a elliptical neighborhood delimitation that extends 5-times further horizontally than vertically. Various orientation angles were tested in order to capture instances of directionally-biased spatial autocorrelation patterns. For instance, a kernel oriented parallel to the container ship would presumably better-capture the spatial autocorrelation pattern and consequently generate more accurate model predictions. An illustration of select kernels tested in the simulations is provided in Figure 5. Upon construction of the anisotropic-kernel spatial

weight matrix, models were with using using *lagsarlm()*, as before.

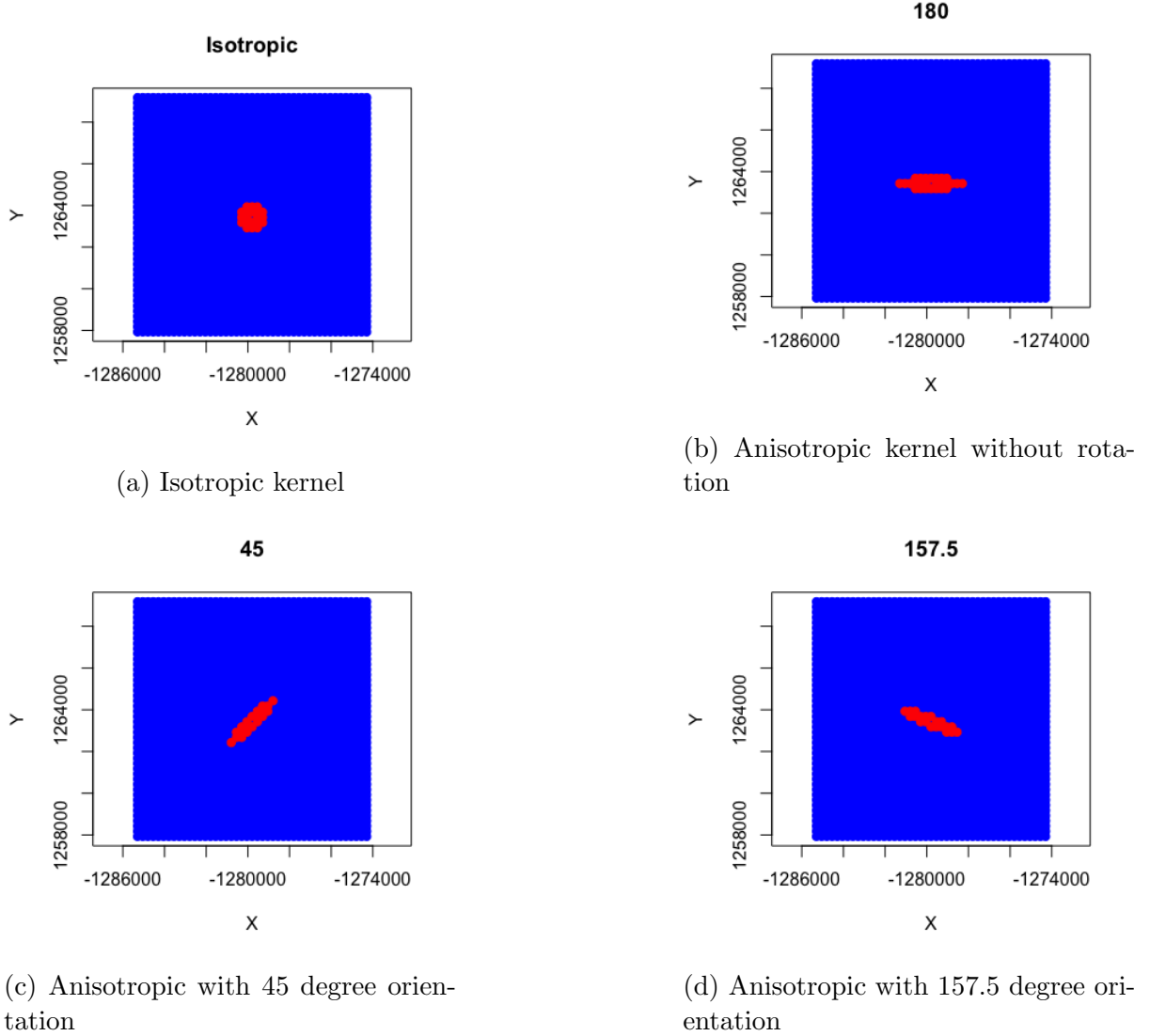


Figure 5: Visualizations of select nearest neighborhood kernels

The final modelling method used was eigenvector spatial filtering. For this, Moran eigenvectors were first determined from the pixel coordinates using the *meigen_f()* function from the *spmoran* package. The resulting eigenvectors were used to estimate the optimal linear eigenvector spatial filtering model using the *esf()* function from *spmoran*. The *esf()* function returns a matrix of a subset of eigenvectors that best explain the spatial dependencies, as determined by AIC-based step-wise selection. Filtering occurs by assigning a cutoff variance inflation factor. The resulting eigenvectors were subsequently included as covaraites in a logistic *glm()* model, consistent with the specifications of the baseline logistic model.

Case Study Results

Burn Area Data Results

Empirical semivariograms for each covariate are presented in Figures 6 and 7 for the first and second burn area data subsets, respectively. For subset 1, the semivariance of each covariate displays a largely-asymptotic behaviour at roughly 1000-2000 m distances (i.e. the *range*), indicating that correlation beyond this lag distance may be insignificant. As such, a sill value of 1000-2000 m may be appropriate. Investigation of the variograms for subset 2 do not reveal as distinct of an asymptotic behaviour. Rather, many of the covariates display a weak-asymptotic leveling at approximately 600-1000m, and then display a linear relationship with lag distance thereafter. Considering these behaviours, bandwidths of 300, 450, 600, and 2000 were employed for the construction of W .

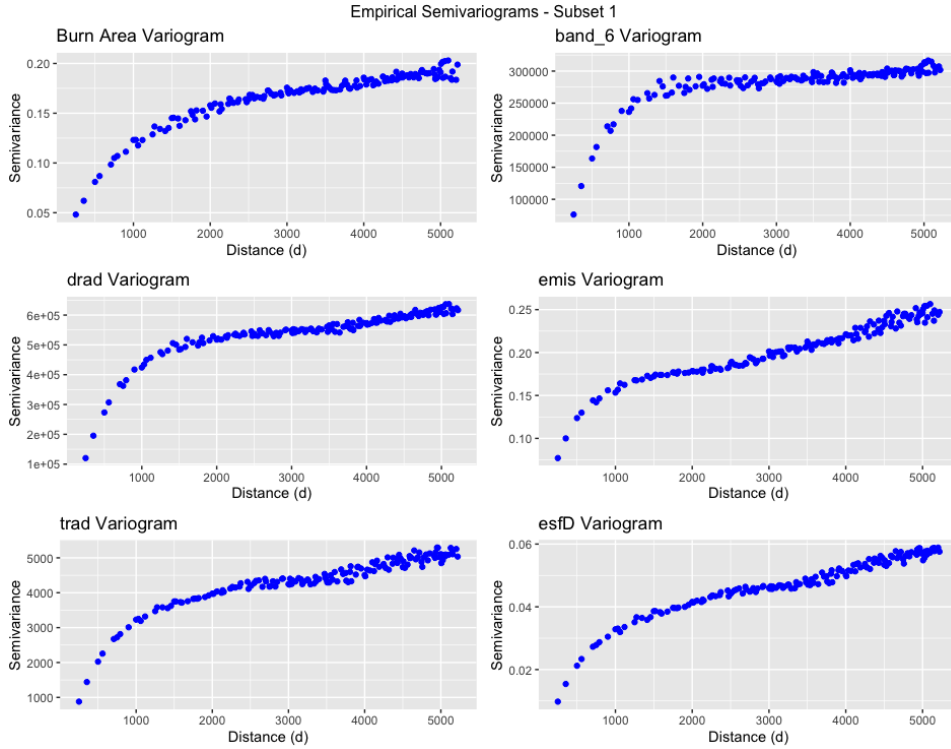


Figure 6: Burn area data subset 1 covariate semivariograms

The resulting burn area model predictions W by nearest neighbor bandwidth are provided in Figures 8 and 9, along with the baseline model predictions and observed values, for comparison. Here a 300 m bandwidth represents a one pixel (i.e. one observation) bandwidth. In both subsets, it is evident that inclusion of a spatial lag with small bandwidths (300 - 450 m) improves model predictions and greatly reduces prediction noise seen in the baseline logistic model. At large bandwidths (i.e. 2000 m), over-fitting to the spatial lag term becomes apparent, as finer-scale spatial variability (i.e. edges) appear overly-smoothed and obscured. This is especially apparent in Subset 1, where several of the observed elongated, narrow burn area corridors are absent from predictions under a 600 m and 2000 m bandwidth.

Burn area predictions using the anisotropic neighborhood spatial weight matrices are presented in Figures 10 and 11. A clear bias towards capturing the corridors that align with the kernel orientation is evident; for instance, the 157.5 degree orientation appears

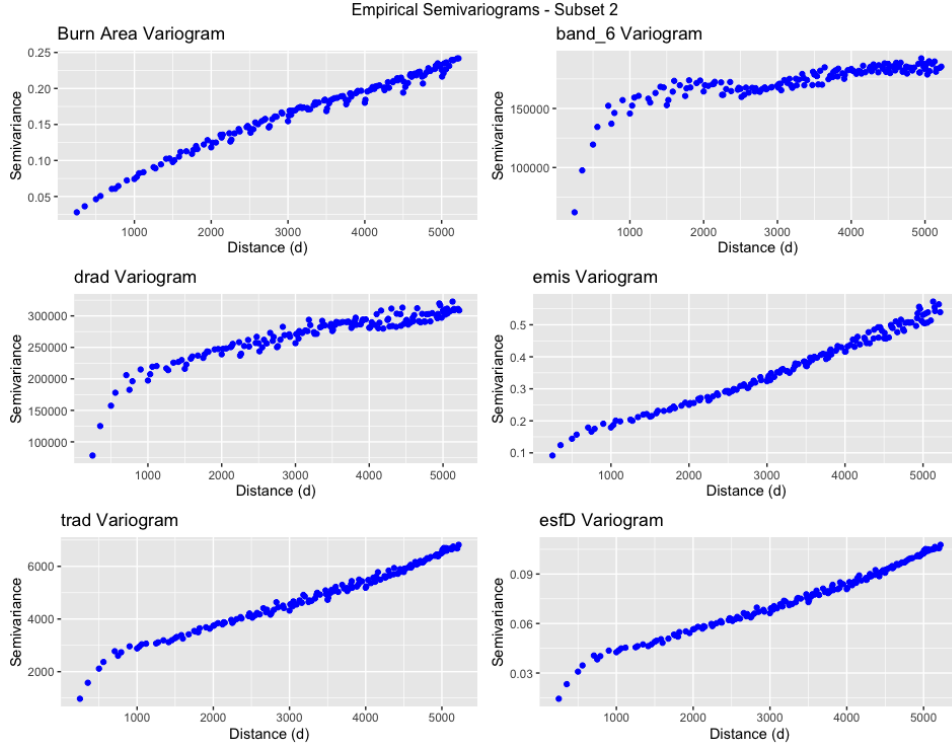


Figure 7: Burn area data subset 2 covariate semivariograms

to capture the near-parallel burn area corridors observed in the northwest portion of Subset 1, whereas these corridors are not present in the predictions using a 90 degree orientation. Despite these effects, predictions appear less accurate than the isotropic 300 m bandwidth kernel, indicating potential over-fitting to the associated directional bias imposed by the anisotropic kernels.

Tabular summaries of burn area model prediction performance are provided in Tables 1 and 2. In both subsets, accounting for spatial autocorrelation using a spatial lag approach improved results. The Moran's I value for the baseline and ESF models were 0.419 and 0.494. Moran's I was greatly reduced and near-zero for the spatial lag models, indicating a lack of residual autocorrelation and an adequate capture of the underlying spatial dependence structure. In subset 1, the lowest Moran's I was encountered in the 450 m bandwidth model, which also yielded the most accurate predictions. In subset 2, the minimum Moran's I was achieved under the 45 degree anisotropic kernel model, but the highest prediction accuracy was achieved in the 600 m bandwidth model. Eigenvector spatial filtering did not meaningfully impact results; prediction accuracy was retained when fitting to the eigenvectors in place of the original covariates. As such, the resulting plots resemble the baseline model and were excluded.

Container Ship Data Results

The empirical semivariogram plots for the container ship data covariates is provided in Figure 12. From the response, it is evident that a range of approximately 25 cells would adequately capture the underlying spatial dependence. Further, the curve exhibits a shape similar to that of a Gaussian or exponential variogram model. Conversely, the semivariogram plots for the red, green, and blue channels are fully linear, and as such, minimal neighborhood bandwidths of 1, 2, 3, and 4 cells were tested.

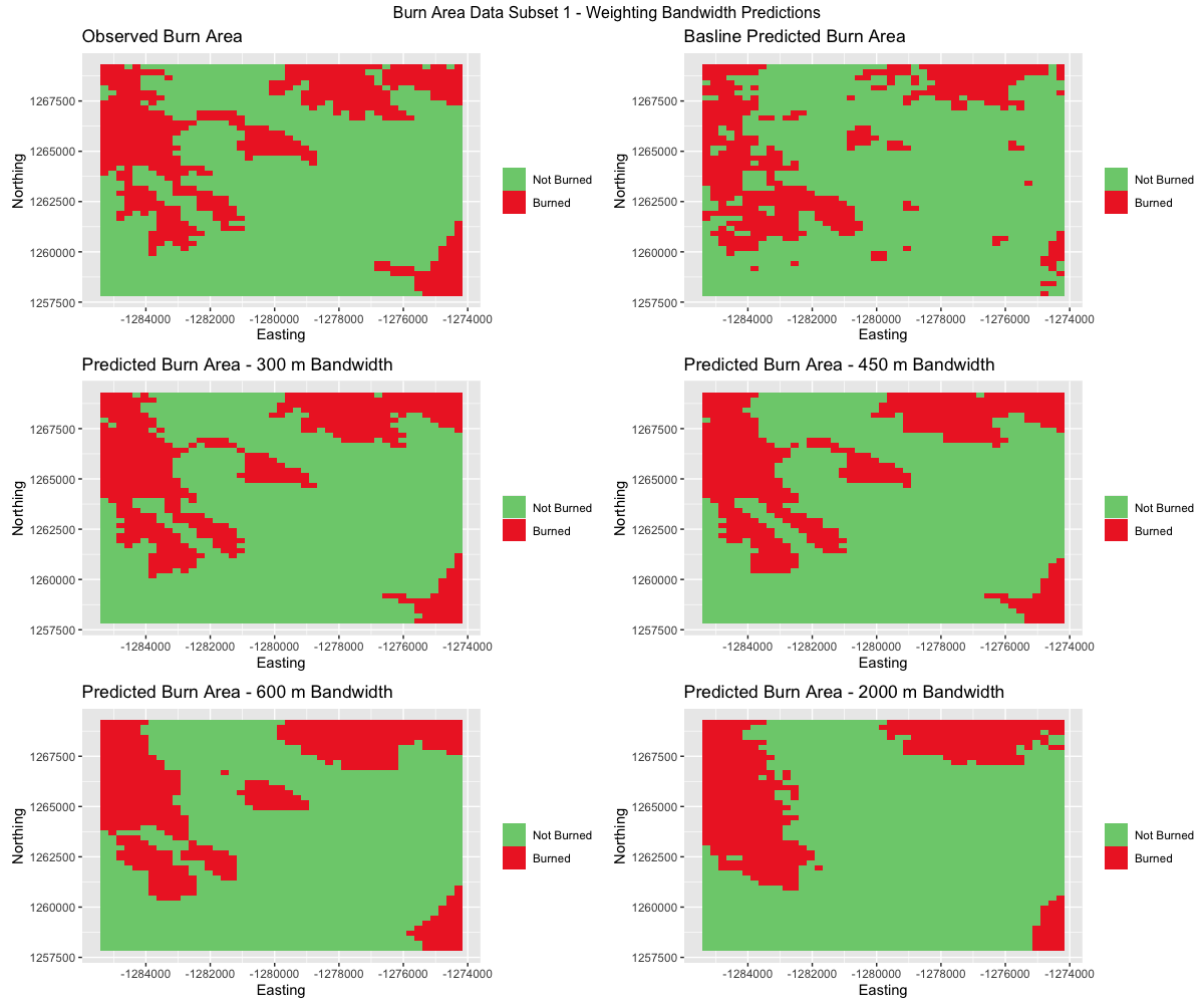


Figure 8: Subset 1 burn area predictions using variable neighborhood bandwidths

Table 1: Burn area data subset 1 - spatial model performance comparison

Model	Accuracy	F1 Score	Resid. Moran's I
Baseline	0.820	0.642	0.419
Bandwidth 300	0.939	0.888	-0.097
Bandwidth 450	0.941	0.891	0.001
Bandwidth 600	0.918	0.847	0.030
Bandwidth 2000	0.861	0.728	0.019
Anisotropic (Angle 0)	0.898	0.811	0.040
Anisotropic (Angle 22.5)	0.886	0.786	0.016
Anisotropic (Angle 45)	0.879	0.768	0.005
Anisotropic (Angle 67.5)	0.888	0.785	0.014
Anisotropic (Angle 90)	0.902	0.816	0.034
Anisotropic (Angle 112.5)	0.908	0.824	0.014
Anisotropic (Angle 135)	0.911	0.831	0.013
Anisotropic (Angle 157.5)	0.909	0.834	0.030
ESF	0.819	0.643	0.422

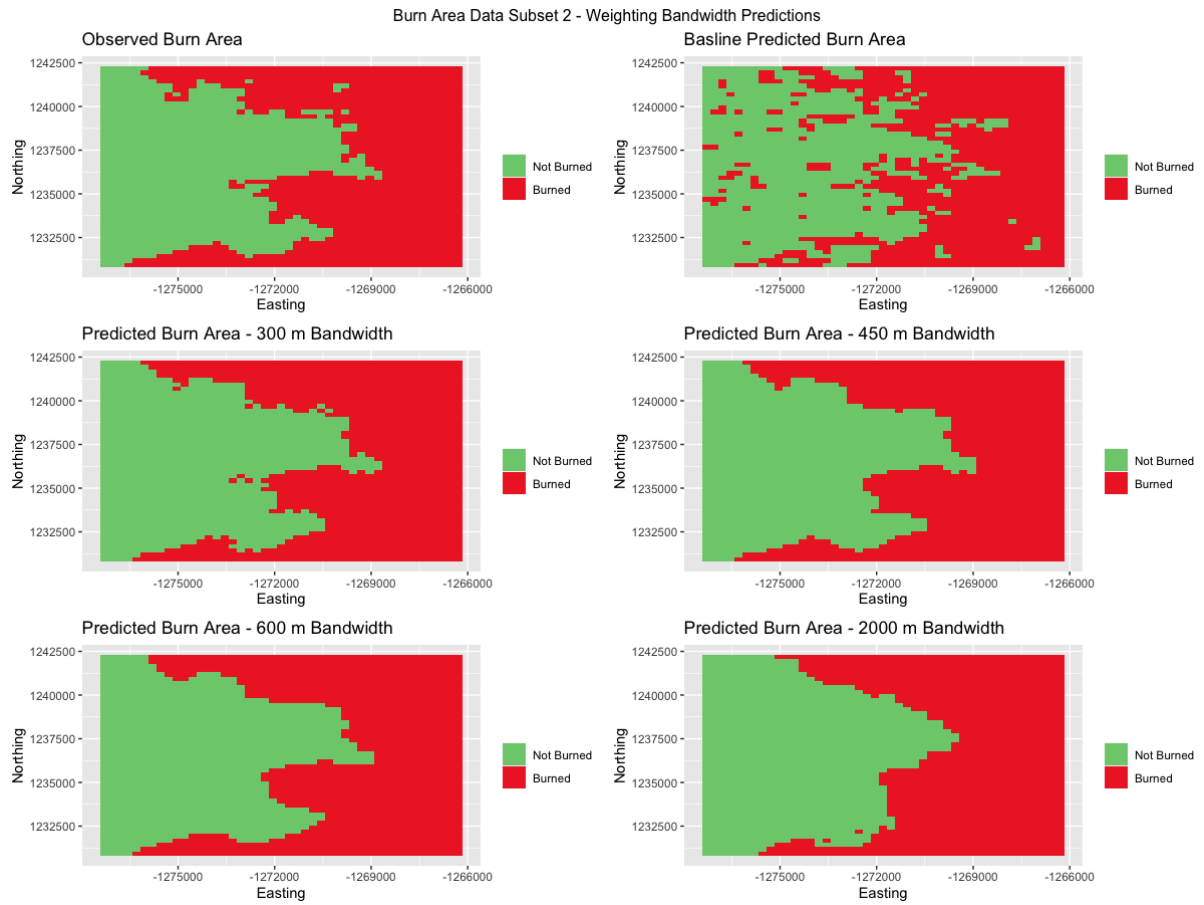


Figure 9: Subset 2 burn area predictions using variable neighborhood bandwidths

Table 2: Burn area data subset 2 - spatial model performance comparison

Model	Accuracy	F1 Score	Resid. Moran's I
Baseline	0.837	0.843	0.494
Bandwidth 300	0.957	0.959	-0.146
Bandwidth 450	0.963	0.964	-0.042
Bandwidth 600	0.960	0.962	0.012
Bandwidth 2000	0.909	0.912	0.055
Anisotropic (Angle 0)	0.955	0.956	0.032
Anisotropic (Angle 22.5)	0.954	0.956	0.022
Anisotropic (Angle 45)	0.951	0.953	0.046
Anisotropic (Angle 67.5)	0.942	0.944	0.047
Anisotropic (Angle 90)	0.946	0.948	0.064
Anisotropic (Angle 112.5)	0.944	0.946	0.027
Anisotropic (Angle 135)	0.954	0.956	0.025
Anisotropic (Angle 157.5)	0.953	0.954	0.025
ESF	0.837	0.843	0.492

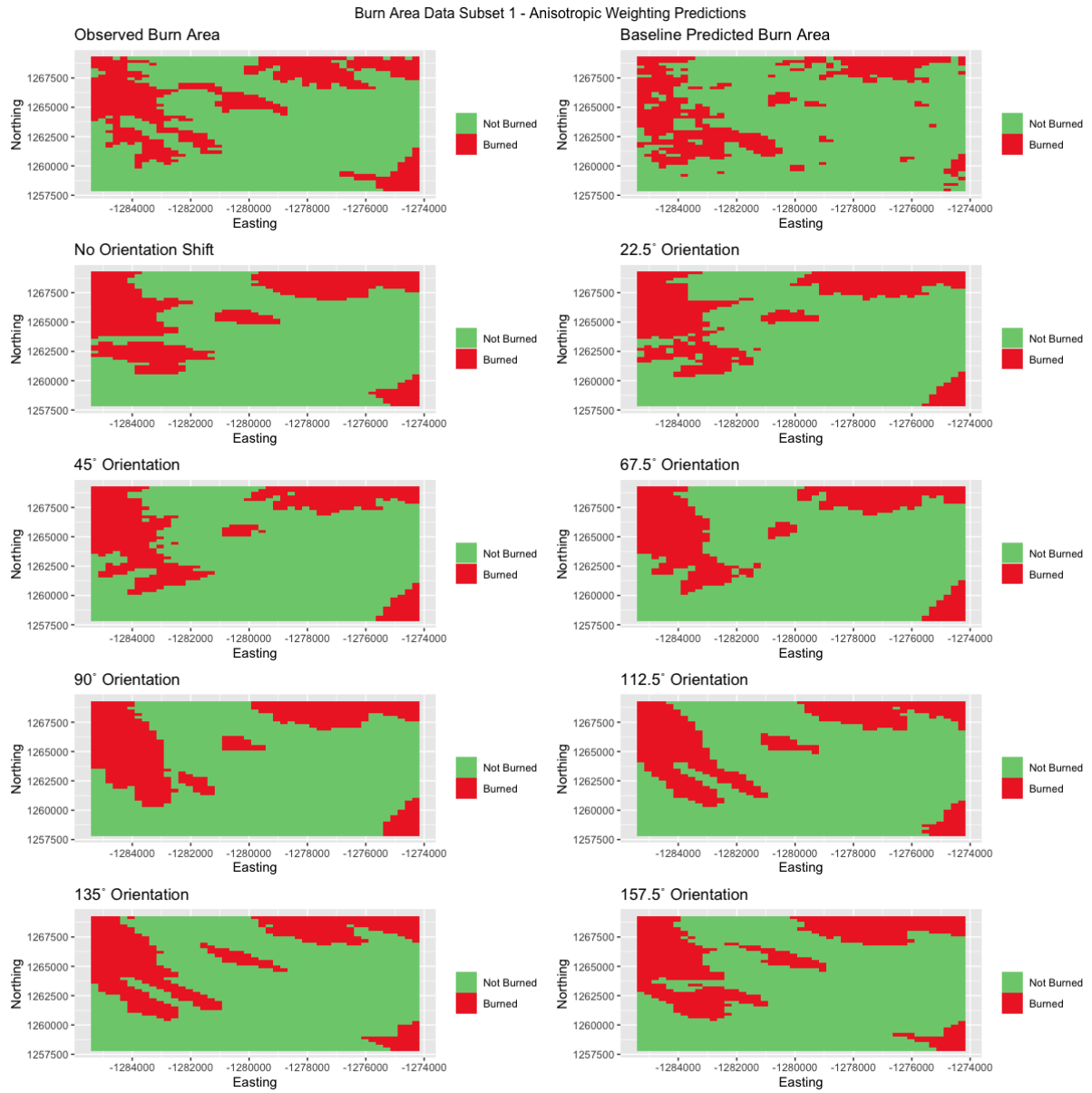


Figure 10: Subset 1 burn area predictions using an anisotropic kernels at various orientations

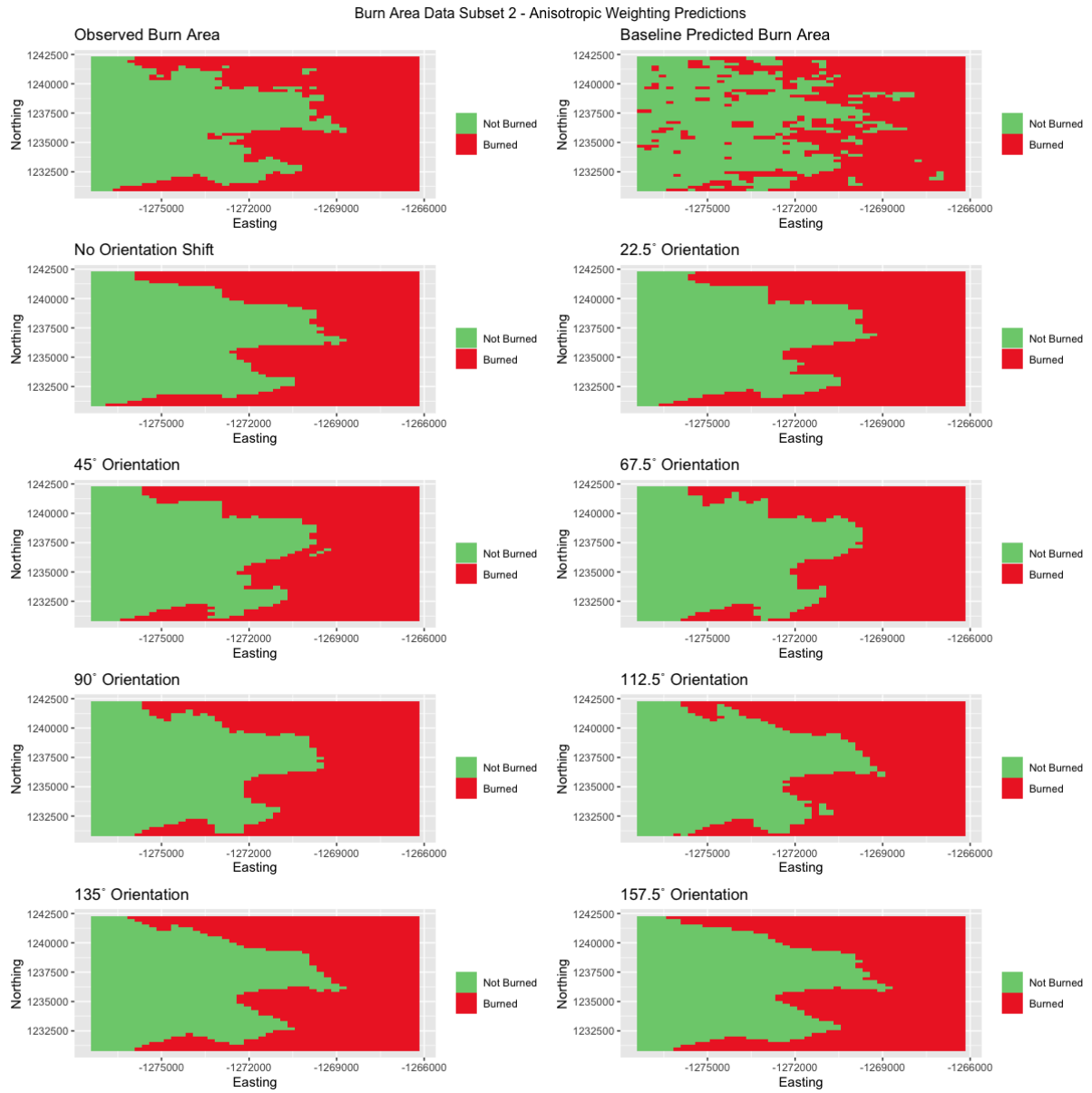


Figure 11: Subset 2 burn area predictions using an anisotropic kernels at various orientations

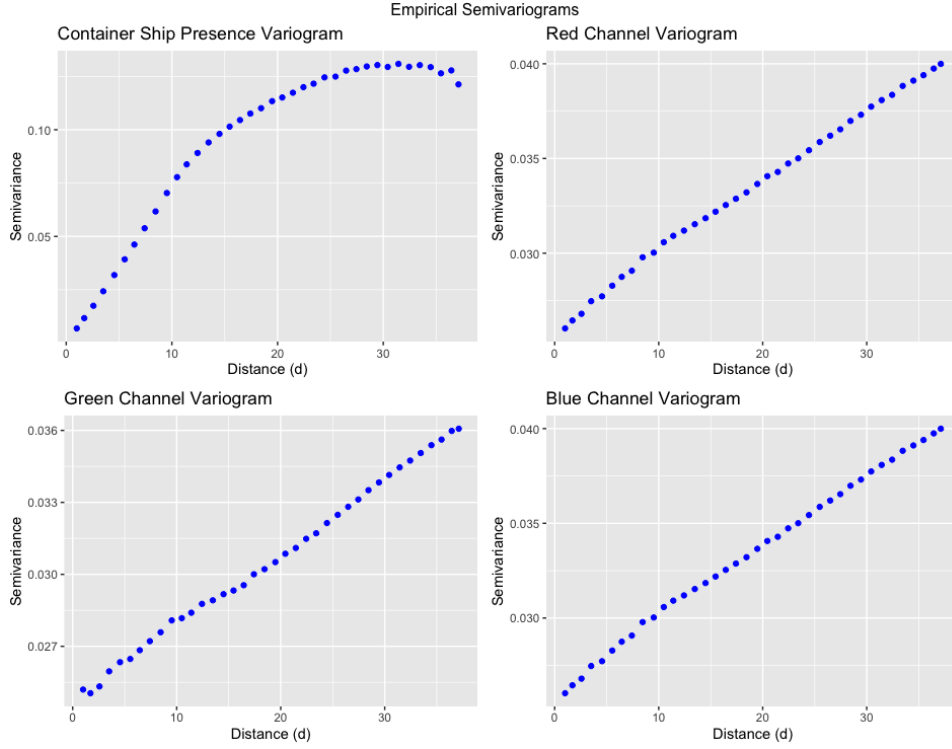


Figure 12: Container ship data covariate semivariograms

The resulting predictions of container ship presence W by nearest neighbor bandwidth are provided in Figure 14, and the predictions using the anisotropic-neighborhoods W by orientation angle are provided in Figure 14. Results follow a pattern analogous to that of the burn area data, wherein a poor-performing baseline model prediction is evidently improved by incorporating a spatial lag. Here, the bandwidth and anisotropic orientation appear to have minimal effect on the overall classification accuracy. Detailed investigation of the classification boundaries show potentially undesirable edge effects (i.e. noise) in orientations that do not align well with the container ship.

A tabular summary of container ship model prediction performance is provided in Table 3. The baseline model, performed poorly in regards to f1-score (and accuracy, relatively). The Moran's I of 0.660 is indicative of significant autocorrelation amongst residuals in the baseline non-spatial model. Near-perfect classification was achieved when incorporating a spatial component. The highest f1-score amongst anisotropic models was achieved in the 45 degree orientation; that which roughly aligns with the orientation of the container ship. Moran's I was lower than 0.1 in absolute value for all spatial models besides the 1-cell bandwidth model.

Discussion

This study sought to review available methods for alleviating spatial autocorrelation amongst residual values in a statistical modelling context. Further, these methods were employed in a practical case study analysis with data used in common applications of spatial modelling. The general process for accounting for spatial autocorrelation, as described in the previous sections, is to first detect and measure it using statistics such as Moran's I or Geary's c . Thereafter, one can investigate potential parameterizations of spatial dependence models (i.e. variograms) by plotting the empirical semi-variograms of

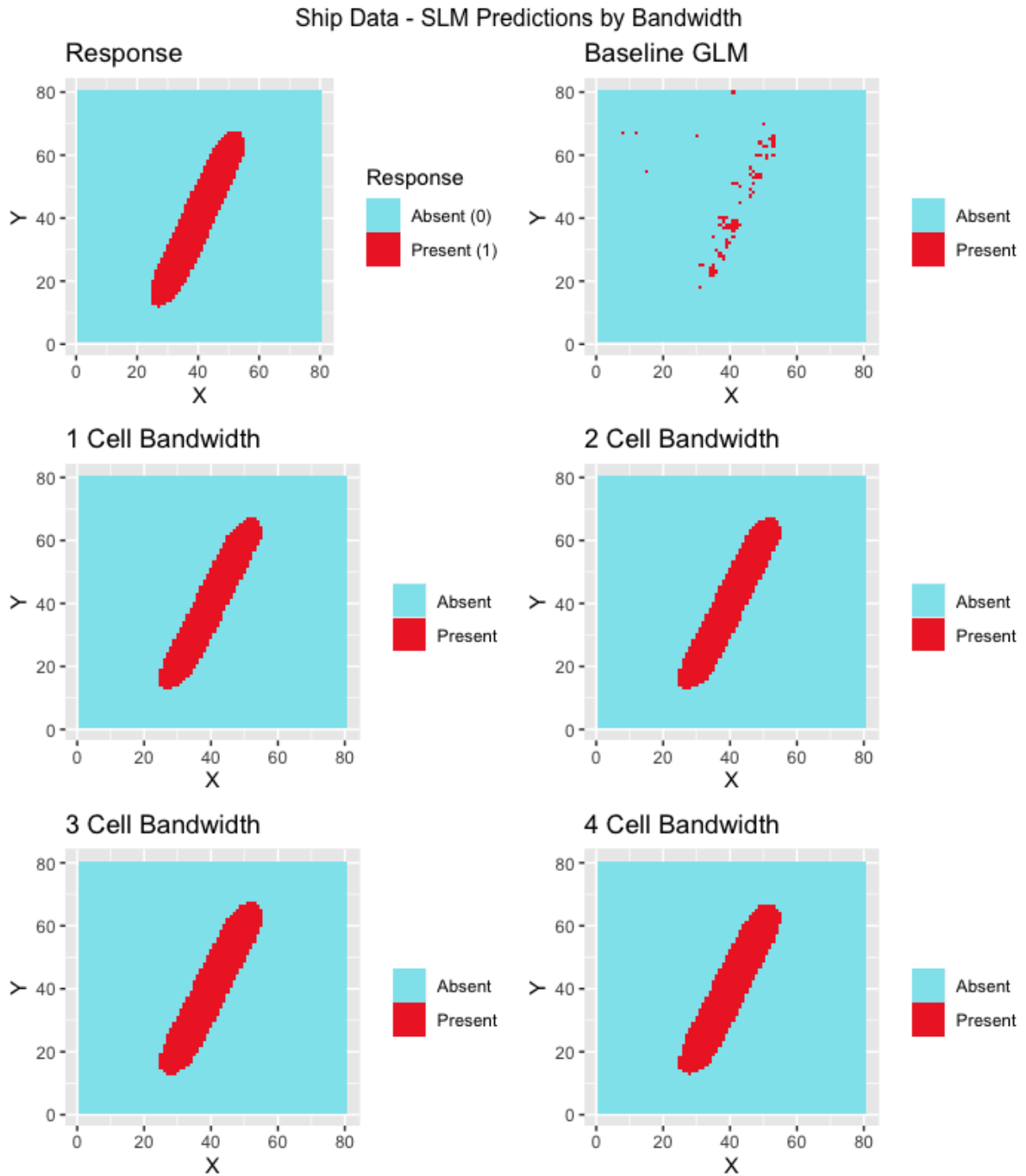


Figure 13: Container ship detection predictions using variable neighborhood bandwidths

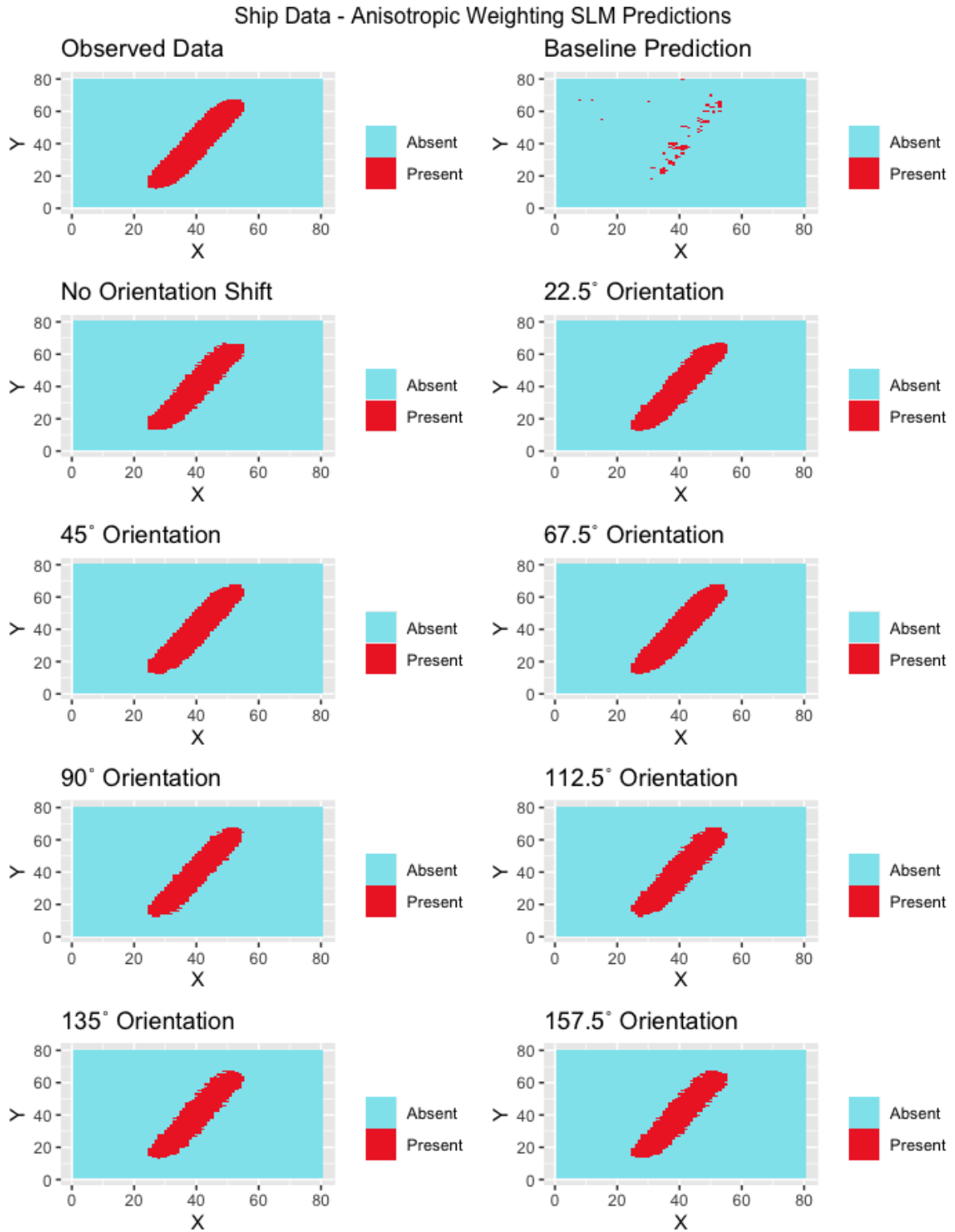


Figure 14: Container ship detection predictions using an anisotropic kernels at various orientations

Table 3: Container ship data - spatial model performance comparison

Model	Accuracy	F1 Score	Resid. Moran's I
Baseline	0.915	0.203	0.660
Bandwidth 1	0.991	0.951	-0.144
Bandwidth 2	0.992	0.958	-0.004
Bandwidth 3	0.998	0.990	0.063
Bandwidth 4	0.997	0.982	0.076
Anisotropic (Angle 0)	0.991	0.951	0.092
Anisotropic (Angle 22.5)	0.993	0.965	0.026
Anisotropic (Angle 45)	0.998	0.989	0.059
Anisotropic (Angle 67.5)	0.998	0.988	0.057
Anisotropic (Angle 90)	0.992	0.955	0.001
Anisotropic (Angle 112.5)	0.991	0.953	0.057
Anisotropic (Angle 135)	0.991	0.951	0.033
Anisotropic (Angle 157.5)	0.991	0.952	0.068
ESF	0.994	0.967	0.057

covariates. From this, one can estimate the range, sill, and nugget coefficients, which are often employed in the construction of theoretical variogram models. Variograms specify weightings of neighboring observations as a function of the distance between them. Once an appropriate theoretical variogram model is specified, a spatial weight matrix can be constructed. This matrix assigns weights, based on the variogram model, to neighboring observations that will presumably influence the value of the central observation. Spatial weight matrices can then be incorporated into spatial lag models to account for the spatial dependence structure and eliminate spatial autocorrelation amongst residuals. Residual spatial autocorrelation can be evaluated using Moran's, and can indicate whether the proposed model adequately captures the underlying spatial dependencies. One can also employ eigenvector spatial filtering as an alternative to spatial lag models, wherein the spatial weighting matrix is decomposed into a subset of eigenvectors that best explain the spatial dependence, and fitting the resulting transformed data to a non-spatial generalized linear model.

The case study completed illustrated the process of accounting for spatial autocorrelation and investigated the effects of various kernel-based methods for constructing the spatial weight matrix. Results were compared to those obtained by non-spatial logistic GLMs, as well as eigenvector spatial filtering models. More specifically, uniformly-weighted distance-based neighborhood kernels (i.e. local average variograms) of varying size, shape, and orientation were employed. The rationale behind employing an anisotropic kernel was to capture directionally-biased patterns of spatial dependence. From the results of the case studies, it is clear that accounting for spatial autocorrelation is vital in the context of satellite image classification analyses, as the baseline results were far worse than that of the spatial lag and ESF models in terms of prediction accuracy, f1-score, and residual Moran's I. In both datasets, employing a bandwidth that captured only the immediately adjacent observation (i.e. above, below, left, and right) was sufficient to generate accurate model predictions. Conversely, an overly-large bandwidth specification appeared to negatively impact model predictions, and provide an overly 'smoothed' representation of the spatial process, especially at classification boundaries.

Employing an anisotropic kernel improved the capture of certain directionally-biased

spatial features that were in the same alignment, but generally did not improve overall model performance. Further, classification boundary noise artifacts were introduced when the kernel was misaligned with the spatial feature. Performance improvements were generally negligible and were only noted in the container ship data, where the anisotropic, directional pattern of spatial features was obvious. It is also noted that incorporating an anisotropic kernel may bias model coefficient estimation. This was illustrated in the case study results; for example, the *blue* RGB band coefficient in the container ship data was estimated at 0.040 in the 157.5 degree orientation, but only 0.008 in the 67.5 degree orientation. Similar patterns were noted in the burn area data, and consequently warrants further examination in future studies.

Considering the results of the case study, it is recommended that such an approach only be taken when there is a clear understanding of directional bias in the underlying spatial dependence structure. Otherwise, a simple radial kernel will suffice to generate accurate spatial classification, and appears robust across the multiple spatial patterns observed in the case study data. Additional applications of anisotropic kernel method could be employed in future studies; for instance, application on the dataset utilized by He et al., 2015, which contains multiple classes and well-formed response class polygon 'clusters' with distinct separation boundaries.

References

- Alberta Geospatial Services. (2023). Alberta Geospatial Services Platform (AGSP) Portal. https://www.alberta.ca/system/files/custom_downloaded_images/fpt-wildfire-accessing-alberta-wildfire-gis-data.pdf
- Banerjee, S. (2015). Hierarchical Modeling and Analysis for Spatial Data, Second Edition.
- Bataineh, A. L., Oswald, B. P., Bataineh, M., Unger, D., Hung, I.-K., & Scognamillo, D. (2006). Spatial autocorrelation and pseudoreplication in fire ecology. *Fire Ecology*, 2(2), 107–118. <https://doi.org/10.4996/fireecology.0202107>
- Chun, Y., Kim, Y., & Campbell, H. (2012). Using Bayesian Methods to Control for Spatial Autocorrelation in Environmental Justice Research: An Illustration Using Toxics Release Inventory Data for a Sunbelt County. *Journal of Urban Affairs*, 34(4), 419–439. <https://doi.org/10.1111/j.1467-9906.2011.00594.x>
- Darmofal, D. (2015). *Spatial analysis for the social sciences*. Cambridge University Press.
- F. Dormann, C., M. McPherson, J., B. Araújo, M., Bivand, R., Bolliger, J., Carl, G., G. Davies, R., Hirzel, A., Jetz, W., Daniel Kissling, W., Kühn, I., Ohlemüller, R., R. Peres-Neto, P., Reineking, B., Schröder, B., M. Schurr, F., & Wilson, R. (2007). Methods to account for spatial autocorrelation in the analysis of species distributional data: A review. *Ecography*, 30(5), 609–628. <https://doi.org/10.1111/j.2007.0906-7590.05171.x>
- Fornango, R. J. (2010). When Space Matters: Spatial Dependence, Diagnostics, and Regression Models. *Journal of Criminal Justice Education*, 21(2), 117–135. <https://doi.org/10.1080/10511251003693652>
- Fortin, M.-J., & Dale, M. R. T. (2005). *Spatial analysis: A guide for ecologists*. Cambridge University Press.
- Getis, A., & Aldstadt, J. (2004). Constructing the Spatial Weights Matrix Using a Local Statistic. *Geographical Analysis*, 36(2), 90–104. <https://doi.org/10.1111/j.1538-4632.2004.tb01127.x>
- Goodchild, M. F. (2013). The quality of big (geo)data [Publisher: SAGE Publications]. *Dialogues in Human Geography*, 3(3), 280–284. <https://doi.org/10.1177/2043820613513392>
- Hammell, R. (2018). Ships in satellite imagery. <https://doi.org/10.34740/KAGGLE/DSV/61115>
- Hawbaker, T. J., Vanderhoof, M. K., Schmidt, G. L., Beal, Y.-J., Picotte, J. J., Takacs, J. D., Falgout, J. T., & Dwyer, J. L. (2020). The Landsat Burned Area algorithm and products for the conterminous United States. *Remote Sensing of Environment*, 244, 111801. <https://doi.org/10.1016/j.rse.2020.111801>
- He, Z., Liu, L., Zhu, Y., & Zhou, S. (2015). Anisotropically foveated nonlocal weights for joint sparse representation-based hyperspectral classification. *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 1–4. <https://doi.org/10.1109/WHISPERS.2015.8075458>
- Kanjir, U., Greidanus, H., & Oštir, K. (2018). Vessel detection and classification from spaceborne optical images: A literature survey. *Remote Sensing of Environment*, 207, 1–26. <https://doi.org/10.1016/j.rse.2017.12.033>
- Kupfer, J. A., & Farris, C. A. (2007). Incorporating spatial non-stationarity of regression coefficients into predictive vegetation models. *Landscape Ecology*, 22(6), 837–852. <https://doi.org/10.1007/s10980-006-9058-2>

- Lam, C., & Souza, P. C. (2020). Estimation and Selection of Spatial Weight Matrix in a Spatial Lag Model. *Journal of Business & Economic Statistics*, 38(3), 693–710. <https://doi.org/10.1080/07350015.2019.1569526>
- Legendre, P. (1993). Spatial Autocorrelation: Trouble or New Paradigm? *Ecology*, 74(6), 1659–1673. <https://doi.org/10.2307/1939924>
- Liu, X., Kounadi, O., & Zurita-Milla, R. (2022). Incorporating Spatial Autocorrelation in Machine Learning Models Using Spatial Lag and Eigenvector Spatial Filtering Features. *ISPRS International Journal of Geo-Information*, 11(4), 242. <https://doi.org/10.3390/ijgi11040242>
- Muhammad, A., Khlood K, A., Salma A, S., Samar O, A., Mashael E, A., Maram A, A., & Maryam, A. (2021). Role of Machine Learning Algorithms in Forest Fire Management: A Literature Review. *Journal of Robotics and Automation*, 5(1). <https://doi.org/10.36959/673/372>
- Pham, B. T., Jaafari, A., Avand, M., Al-Ansari, N., Dinh Du, T., Yen, H. P. H., Phong, T. V., Nguyen, D. H., Le, H. V., Mafi-Gholami, D., Prakash, I., Thi Thuy, H., & Tuyen, T. T. (2020). Performance Evaluation of Machine Learning Methods for Forest Fire Modeling and Prediction. *Symmetry*, 12(6), 1022. <https://doi.org/10.3390/sym12061022>
- Ploton, P., Mortier, F., Réjou-Méchain, M., Barbier, N., Picard, N., Rossi, V., Dormann, C., Cornu, G., Viennois, G., Bayol, N., Lyapustin, A., Gourlet-Fleury, S., & Pélissier, R. (2020). Spatial validation reveals poor predictive performance of large-scale ecological mapping models. *Nature Communications*, 11(1), 4540. <https://doi.org/10.1038/s41467-020-18321-y>
- Portier, J., Gauthier, S., Robitaille, A., & Bergeron, Y. (2018). Accounting for spatial autocorrelation improves the estimation of climate, physical environment and vegetation’s effects on boreal forest’s burn rates. *Landscape Ecology*, 33(1), 19–34. <https://doi.org/10.1007/s10980-017-0578-8>
- QGIS. (2023). QGIS, A Free and Open Source Geographic Information System. Retrieved October 31, 2023, from <https://qgis.org/en/site/>
- Stakhovych, S., & Bijmolt, T. H. (2009). Specification of spatial models: A simulation study on weights matrices. *Papers in Regional Science*, 88(2), 389–409. <https://doi.org/10.1111/j.1435-5957.2008.00213.x>
- Tobler, W. R. (1970). A Computer Movie Simulating Urban Growth in the Detroit Region [Publisher: [Clark University, Wiley]]. *Economic Geography*, 46, 234–240. <https://doi.org/10.2307/143141>
- US Geological Survey. (2023). Landsat Science Products — U.S. Geological Survey. Retrieved October 31, 2023, from <https://www.usgs.gov/landsat-missions/landsat-science-products>
- Van Lieshout, M. N. M. (2019). *Theory of spatial statistics: A concise introduction*. CRC Press, Taylor & Francis Group.
- Yu, W. (2010). Constructing Spatial Weight Matrix Using Local Spatial Statistics And Its Applications.

Appendix A: R Scripts

Burn Area Data R Script

```
library(sp)
library(spdep)
library(spatialreg)
library(stats)
library(MASS)
library(spgwr)
library(ggplot2)
library(gridExtra)
library(grid)
library(gstat)
library(spmoran)
library(spfilter)
library(tidyverse)
library(caret)
library(pROC)

raw_data <- read.csv('landsat_data_final.csv',
                    colClasses=c(rep("numeric", 13)))

# process response
raw_data$burn_area <- ifelse(raw_data$burn_area == 65535, 0, 1)

par(mfrow = c(1, 1))
# pairs(raw_data)

subset_scale <- 0.45
# subset 2
# shift_x <- -8000
# shift_y <- -7000
# subset 1
shift_x <- -16000
shift_y <- 20000
x_range <- max(raw_data$X) - min(raw_data$X)
x_lower <- min(raw_data$X) + x_range*subset_scale + shift_x
x_upper <- min(raw_data$X) + x_range*(1-subset_scale) + shift_x

y_range <- max(raw_data$Y) - min(raw_data$Y)
y_lower <- min(raw_data$Y) + y_range*subset_scale + shift_y
y_upper <- min(raw_data$Y) + y_range*(1-subset_scale) + shift_y

data <- raw_data[raw_data$X >= x_lower & raw_data$X <= x_upper & raw_data$Y >= y_lower, ]
data <- data[c("X", "Y", "burn_area", "band_6", "drad", "emis", "emsd", "trad")]

plot(data$X, data$Y, col=data$burn_area)
```

```

# defining spatial weights matrix
coords <- as.matrix(coordinates(data[, c("X", "Y")]))
W <- dnearneigh(coords, 0, 300)
W <- nb2listw(W)

# Moran test
moran.test(data$burn_area, W)
moran.test(data$band_6, W)
moran.test(data$trad, W)
moran.test(data$drad, W)
moran.test(data$emis, W)

par(mfrow = c(1, 1))
burned_colors <- c('palegreen3', 'firebrick2')
plot_raw <- ggplot(data, aes(x=X, y=Y, fill=as.factor(burn_area))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Observed Burn Area', x='Easting', y='Northing', fill='')

plot_band6 <- ggplot(data, aes(x=X, y=Y, fill=band_6)) +
  geom_tile() +
  labs(title = 'Band 6', x='Easting', y='Northing', fill='')

plot_drad <- ggplot(data, aes(x=X, y=Y, fill=drad)) +
  geom_tile() +
  labs(title = 'drad', x='Easting', y='Northing', fill='')

plot_emis <- ggplot(data, aes(x=X, y=Y, fill=emis)) +
  geom_tile() +
  labs(title = 'emis', x='Easting', y='Northing', fill='')

plot_trad <- ggplot(data, aes(x=X, y=Y, fill=trad)) +
  geom_tile() +
  labs(title = 'trad', x='Easting', y='Northing', fill='')

plot_emsd <- ggplot(data, aes(x=X, y=Y, fill=emsd)) +
  geom_tile() +
  labs(title = 'emsd', x='Easting', y='Northing', fill='')

grid.arrange(plot_raw, plot_band6, plot_drad,
              plot_emis, plot_trad, plot_emsd,
              ncol=2, nrow=3, top=textGrob("Burn Area Data Covariates - Subset 1"))

####-----####
# baseline non-spatial model

```

```

####-----####
formula <- burn_area ~ band_6 + drad + emis + trad # + emsd
logit_baseline <- glm(formula, data = data, family = binomial)
summary(logit_baseline)
results_baseline <- data.frame(X=data$X, Y=data$Y, fitted=logit_baseline$fitted.values)
moran_i_baseline <- moran.test(residuals(logit_baseline), listw =W)

####-----####
# spatial logistic models
####-----####

# defining inverse log link function
log_link <- function(x) {
  log(x / (1 - x))
}

# defining spatial weights matrix
coords <- as.matrix(coordinates(data[, c("X", "Y")]))
W <- dnearneigh(coords, 0, 300)
W <- nb2listw(W)

# comparing bandwidths
bandwidths <- c(300, 450, 600, 2000)
results_bandwidths <- data.frame(X=data$X, Y=data$Y)
results_bw_coefs <- data.frame()
results_bw_aic <- data.frame()
results_bw_moran <- data.frame()
for (bandwidth in bandwidths){
  W <- dnearneigh(coords, 0, bandwidth)
  W <- nb2listw(W)
  # spatial lag model (linear predictor)
  slm_model <- lagsarlm(burn_area ~ band_6 + drad + emis + trad,
                        data=data,
                        listw=W)
  # link linear predictor
  y_prob <- ifelse(slm_model$fitted.values > 1, 1, slm_model$fitted.values)
  y_prob <- ifelse(y_prob < 0, 0, y_prob)
  y_prob <- plogis(log_link(y_prob))
  y_pred <- ifelse(y_prob > 0.5, 1, 0)

  # save predictions
  results_bandwidths[, paste0("bandwidth_", bandwidth)] <- y_pred

  # save coefficient, AIC, and morans I results
  results_bw_coefs <- rbind(results_bw_coefs, coef(summary(slm_model)))
  results_bw_aic <- rbind(results_bw_aic,
                          data.frame(bw=bandwidth[1],

```

```

aicc=slm_model[["AIC_lm.model"]]))
moran_i <- moran.test(residuals(slm_model), listw=W)
results_bw_moran <- rbind(results_bw_moran,
                           data.frame(bw = bandwidth[1],
                                       Moran_I = moran_i[["estimate"]][["Moran I
                                       Exp = moran_i[["estimate"]][["Expectation
                                       Var = moran_i[["estimate"]][["Expectation
}

### GGLOTS ###

par(mfrow = c(1, 1))
burned_colors <- c('palegreen3', 'firebrick2')
plot_raw <- ggplot(data, aes(x=X, y=Y, fill=as.factor(burn_area))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Observed Burn Area', x='Easting', y='Northing', fill='')

plot_baseline <- ggplot(results_baseline, aes(x=X, y=Y, fill=as.factor(round(fitted))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Baseline Predicted Burn Area', x='Easting', y='Northing', fill='')

plot_bw300 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_300))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Predicted Burn Area - 300 m Bandwidth', x='Easting', y='Northing', fi

plot_bw450 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_450))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Predicted Burn Area - 450 m Bandwidth', x='Easting', y='Northing', fi

plot_bw600 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_600))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Predicted Burn Area - 600 m Bandwidth', x='Easting', y='Northing', fi

plot_bw750 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_2000))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Predicted Burn Area - 2000 m Bandwidth', x='Easting', y='Northing', f

grid.arrange(plot_raw, plot_baseline, plot_bw300,
              plot_bw450, plot_bw600, plot_bw750,
              ncol=2, nrow=3, top=textGrob("Burn Area Data Subset 1 - Weighting Bandwi

```

```

####-----####
# anisotropic variogram/kernel
####-----####

scale_x <- 0.2
scale_y <- 1
rotations <- c(180, 22.5, 45, 67.5, 90, 112.5, 135, 157.5)
results_kernels <- data.frame(X=data$X, Y=data$Y)
results_kernels_coefs <- data.frame()
results_kernels_aic <- data.frame()
results_kernels_moran <- data.frame()

for (angle in rotations){

  rot_mtx <- matrix(c(cos(angle*(pi/180)), sin(angle*(pi/180)),
                    -sin(angle*(pi/180)), cos(angle*(pi/180))),
                  nrow=2)

  rotated_coords <- coords %*% rot_mtx
  rotated_coords <- as.matrix(coordinates(rotated_coords))

  scaled_coords <- as.matrix(data.frame(
    X = rotated_coords[,1] * scale_x,
    Y = rotated_coords[,2] * scale_y
  ))

  W <- dnearneigh(scaled_coords, 0, 300) # 300 distance threshold neighborhood
  W <- nb2listw(W)

  # plot kernel - update
  middle <- unlist(W$neighbours[1035])
  neighbor_points <- data[middle,]
  plot(data[,1], data[,2], pch=19,
       col="blue", xlab = "X", ylab = "Y",
       main = angle,
       asp=1)
  points(neighbor_points[,1], neighbor_points[,2],
        pch=19, col="red")
  #linear predictor
  slm_model <- lagsarlm(burn_area ~ band_6 + drad + emis + trad,
                      data=data,
                      listw=W)
  #constrain and transform linear predictor with log link
  y_prob <- ifelse(slm_model$fitted.values > 1, 1, slm_model$fitted.values)
  y_prob <- ifelse(y_prob < 0, 0, y_prob)
  y_prob <- plogis(log_link(y_prob))
  y_pred <- ifelse(y_prob > 0.5, 1, 0)
  results_kernels[, paste0("angle_", angle)] <- y_pred
}

```

```

# save coefficient, AIC, and morans I results
results_kernels_coefs <- rbind(results_kernels_coefs, coef(summary(slm_model)))
results_kernels_aic <- rbind(results_kernels_aic,
                             data.frame(angle=angle[1],
                                           aic=slm_model[["AIC_lm.model"]]))
moran_i <- moran.test(residuals(slm_model), listw=W)
results_kernels_moran <- rbind(results_kernels_moran,
                                data.frame(angle = angle[1],
                                            Moran_I = moran_i[["estimate"]][["Moran I
                                            Exp = moran_i[["estimate"]][["Expectation
                                            Var = moran_i[["estimate"]][["Expectation

}

### PLOTS ####

par(mfrow = c(1, 1))
burned_colors <- c('palegreen3', 'firebrick2')
plot_raw <- ggplot(data, aes(x=X, y=Y, fill=as.factor(burn_area))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Observed Burn Area', x='Easting', y='Northing', fill='')

plot_baseline <- ggplot(results_baseline, aes(x=X, y=Y, fill=as.factor(round(fitted)))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Baseline Predicted Burn Area', x='Easting', y='Northing', fill='')

plot_x00 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_180')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'No Orientation Shift', x='Easting', y='Northing', fill='')

plot_x0 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_22.5')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '22.5 Orientation', x='Easting', y='Northing', fill='')

plot_x1 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_45')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '45 Orientation', x='Easting', y='Northing', fill='')

plot_x2 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_67.5')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '67.5 Orientation', x='Easting', y='Northing', fill='')

```

```

plot_x3 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_90')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '90 Orientation', x='Easting', y='Northing', fill=''))

plot_x4 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_112.5')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '112.5 Orientation', x='Easting', y='Northing', fill=''))

plot_x5 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_135')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '135 Orientation', x='Easting', y='Northing', fill=''))

plot_x6 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_157.5')) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = '157.5 Orientation', x='Easting', y='Northing', fill=''))

grid.arrange(plot_raw, plot_baseline, plot_x00, plot_x0,
  plot_x1, plot_x2, plot_x3,
  plot_x4, plot_x5, plot_x6,
  ncol=2, nrow=5, top=textGrob("Burn Area Data Subset 1 - Anisotropic Weig

####-----####
# ESF logistic models
####-----####

# using spmoran
# https://rdr.io/cran/spmoran/man/esf.html
meig_f<- meigen_f(coords=coords)

# vif 10 cutoff
esfD <- esf(y=data$burn_area, x=data[4:8],
  meig=meig_f,
  vif=10,
  fn="AIC")
transformed_data <- data.frame(esfD$pred, data)
logit_esf_10 <- glm(burn_area ~ esfD.pred,# band_6 + drad + emis + trad,
  data = transformed_data,
  family = binomial)

# compile
results_esf <- data.frame(X=data$X, Y=data$Y,
  fitted_20=logit_esf_10$fitted.values)

```

```

par(mfrow = c(1, 1))
ggplot(results_esf, aes(x=X, y=Y, fill=as.factor(round(fitted_20)))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Not Burned', 'Burned')) +
  labs(title = 'Eigenvector Spatial Filter GLM Predictions - Subset 2', x='Easting', y='Northing')

####-----####
# empirical variogram plots
####-----####

burn_area <- SpatialPointsDataFrame(coords=coords,
                                   data=data.frame(value=data$burn_area))
emp_variogram <- variogram(value ~ 1, data = burn_area, width = 10)
burn_area_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "Burn Area Variogram",
       x = "Distance (d)", y = "Semivariance")

band_6 <- SpatialPointsDataFrame(coords=coords,
                                data=data.frame(value=data$band_6))
emp_variogram <- variogram(value ~ 1, data = band_6, width = 10)
band_6_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "band_6 Variogram",
       x = "Distance (d)", y = "Semivariance")

drad <- SpatialPointsDataFrame(coords=coords,
                              data=data.frame(value=data$drad))
emp_variogram <- variogram(value ~ 1, data = drad, width = 10)
drad_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "drad Variogram",
       x = "Distance (d)", y = "Semivariance")

emis <- SpatialPointsDataFrame(coords=coords,
                              data=data.frame(value=data$emis))
emp_variogram <- variogram(value ~ 1, data = emis, width = 10)
emis_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "emis Variogram",
       x = "Distance (d)", y = "Semivariance")

trad <- SpatialPointsDataFrame(coords=coords,
                              data=data.frame(value=data$trad))
emp_variogram <- variogram(value ~ 1, data = trad, width = 10)
trad_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +

```

```

    geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
    labs(title = "trad Variogram",
         x = "Distance (d)", y = "Semivariance")

esfD_pred <- SpatialPointsDataFrame(coords=coords,
                                   data=data.frame(value=esfD$pred))
emp_variogram <- variogram(value ~ 1, data = esfD_pred, width = 10)
esfD_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "esfD Variogram",
       x = "Distance (d)", y = "Semivariance")

grid.arrange(burn_area_vgm_plot,
              band_6_vgm_plot,
              drad_vgm_plot,
              emis_vgm_plot,
              trad_vgm_plot,
              esfD_vgm_plot,
              ncol=2, nrow=3,
              top=textGrob("Empirical Semivariograms - Subset 2"))

####-----####
# model performance comparison
####-----####

# prediction accuracy
# baseline
mean(data$burn_area == round(results_baseline$fitted))
# bandwidths
mean(data$burn_area == results_bandwidths$bandwidth_300)
mean(data$burn_area == results_bandwidths$bandwidth_450)
mean(data$burn_area == results_bandwidths$bandwidth_600)
mean(data$burn_area == results_bandwidths$bandwidth_2000)
# anisotropic kernel accuary
mean(data$burn_area == results_kernels$angle_180)
mean(data$burn_area == results_kernels$angle_22.5)
mean(data$burn_area == results_kernels$angle_45)
mean(data$burn_area == results_kernels$angle_67.5)
mean(data$burn_area == results_kernels$angle_90)
mean(data$burn_area == results_kernels$angle_112.5)
mean(data$burn_area == results_kernels$angle_135)
mean(data$burn_area == results_kernels$angle_157.5)
# esf
mean(data$burn_area == round(results_esf$fitted_20))

#f1 scores
#baseline
precision <- posPredValue(as.factor(round(results_baseline$fitted)),

```

```

                                as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(round(results_baseline$fitted)),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

#bands
precision <- posPredValue(as.factor(results_bandwidths$bandwidth_300),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_300),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_450),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_450),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_600),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_600),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_2000),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_2000),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

# anisotropic
precision <- posPredValue(as.factor(results_kernels$angle_180),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_180),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_22.5),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_22.5),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_45),
                        as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_45),
                    as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

```

```

precision <- posPredValue(as.factor(results_kernels$angle_67.5),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_67.5),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_90),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_90),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_112.5),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_112.5),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_135),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_135),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_157.5),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_157.5),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

# esf
precision <- posPredValue(as.factor(round(results_esf$fitted_20)),
                          as.factor(data$burn_area), positive="1")
recall <- sensitivity(as.factor(round(results_esf$fitted_20)),
                     as.factor(data$burn_area), positive="1")
(2 * precision * recall) / (precision + recall)

# morans I
moran.test(residuals(logit_baseline), listw=W)
round(results_bw_moran$Moran_I, 3)
round(results_kernels_moran$Moran_I, 3)
moran.test(residuals(logit_esf_10), listw=W)

```

Container Ship Data R Script

```
library(png)
```

```

library(pracma)
library(ggplot2)
library(gridExtra)
library(grid)
library(spdep)
library(sp)
library(spatialreg)

# https://stackoverflow.com/questions/14769628/how-to-get-pixel-data-from-an-image-us
img_class_path <- "/Users/johntweedie/dev/gis/data/ships/class.png"
img_class <- round(readPNG(img_class_path))[, , 1]

img_train_path <- "/Users/johntweedie/dev/gis/data/ships/train3.png"
# img_train_path <- "/Users/johntweedie/dev/gis/data/ships/train2.png"
# img_train_path <- "/Users/johntweedie/dev/gis/data/ships/train.png"
img_train <- readPNG(img_train_path)[, , ]
img_red <- img_train[, , 1]
img_green <- img_train[, , 2]
img_blue <- img_train[, , 3]
# hist(img_red)
# hist(img_green)
# hist(img_blue)

# extract picture RGB components and put into dataframe
data <- list()
for (x in seq_len(dim(img_red)[2])){
  for (y in seq_len(dim(img_red)[1])){
    data[[length(data) + 1]] <- data.frame(X=x, Y=y,
                                             red=img_red[y, x],
                                             green=img_green[y, x],
                                             blue=img_blue[y, x],
                                             class=img_class[y, x])
  }
}
data <- do.call(rbind, data)
burned_colors <- c('cadetblue2', 'firebrick2')
class_plot <- ggplot(data, aes(x=X, y=Y, fill=as.factor(class))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent (0)', 'Present (1)')) +
  labs(title = 'Response', x='X', y='Y', fill='Response')

red_plot <- ggplot(data, aes(x=X, y=Y, fill=red)) +
  geom_tile() +
  labs(title = 'Red Channel', x='X', y='Y', fill='') +
  scale_fill_gradient(low = "white", high = "red")

green_plot <- ggplot(data, aes(x=X, y=Y, fill=green)) +
  geom_tile() +

```

```

labs(title = 'Green Channel', x='X', y='Y', fill='') +
scale_fill_gradient(low = "white", high = "green")

blue_plot <- ggplot(data, aes(x=X, y=Y, fill=blue)) +
  geom_tile() +
  labs(title = 'Blue Channel', x='X', y='Y', fill='') +
  scale_fill_gradient(low = "white", high = "blue")

grid.arrange(class_plot, red_plot, green_plot, blue_plot,
              ncol=2, nrow=2, top=textGrob("Ship Image Data Variables"))

####-----####
# baseline non-spatial model
####-----####
formula <- class ~ red + green + blue
logit_baseline <- glm(formula, data = data, family = binomial)
summary(logit_baseline)
results_baseline <- data.frame(X=data$X, Y=data$Y, fitted=round(logit_baseline$fitted
plot(results_baseline$X, results_baseline$Y, col=round(results_baseline$fitted))

####-----####
# spatial logistic models
####-----####
# spatials weights matrix
coords <- as.matrix(coordinates(data[, c("X", "Y")]))
W <- dnearneigh(coords, 0, 3)
W <- nb2listw(W)

# Spatial autocorrelation test
red <- moran.test(data$red, W)
green <- moran.test(data$green, W)
blue <- moran.test(data$blue, W)
class <- moran.test(data$class, W)

log_link <- function(x) {
  log(x / (1 - x))
}

# spatial lag model - alternate bandwidths for W
bandwidths <- c(1, 2, 3, 4)
results_bandwidths <- data.frame(X=data$X, Y=data$Y)
results_bw_coefs <- data.frame()
results_bw_aic <- data.frame()
results_bw_moran <- data.frame()

```

```

for (bandwidth in bandwidths){
  W <- dnearneigh(coords, 0, bandwidth)
  W <- nb2listw(W)
  slm_model <- lagsarlm(formula,
                        data=data,
                        listw=W)
  y_prob <- ifelse(slm_model$fitted.values > 1, 1, slm_model$fitted.values)
  y_prob <- ifelse(y_prob < 0, 0, y_prob)
  y_prob <- plogis(log_link(y_prob))
  y_pred <- ifelse(y_prob > 0.5, 1, 0)
  results_bandwidths[, paste0("bandwidth_", bandwidth)] <- y_pred
  # save coefficient, AIC, and morans I results

  results_bw_coefs <- rbind(results_bw_coefs, coef(summary(slm_model)))
  results_bw_aic <- rbind(results_bw_aic,
                          data.frame(bw=bandwidth[1],
                                      aic=slm_model[["AIC_lm.model"]]))
  moran_i <- moran.test(residuals(slm_model), listw=W)
  results_bw_moran <- rbind(results_bw_moran,
                            data.frame(bw = bandwidth[1],
                                        Moran_I = moran_i[["estimate"]][["Moran I stat
                                        Exp = moran_i[["estimate"]][["Expectation"]],
                                        Var = moran_i[["estimate"]][["Expectation"]]))
}

# results plotting
baseline_plot <- ggplot(results_baseline, aes(x=X, y=Y, fill=as.factor(fitted))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = 'Baseline GLM', x='X', y='Y', fill='')

slm_plot1 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_1))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '1 Cell Bandwidth', x='X', y='Y', fill='')

slm_plot2 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_2))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '2 Cell Bandwidth', x='X', y='Y', fill='')

slm_plot3 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_3))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '3 Cell Bandwidth', x='X', y='Y', fill='')

slm_plot4 <- ggplot(results_bandwidths, aes(x=X, y=Y, fill=as.factor(bandwidth_4))) +

```

```

geom_tile() +
scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
labs(title = '4 Cell Bandwidth', x='X', y='Y', fill='')

grid.arrange(class_plot, #red_plot, green_plot, blue_plot,
              baseline_plot, slm_plot1, slm_plot2, slm_plot3, slm_plot4,
              ncol=2, nrow=3, top=textGrob("Ship Data - SLM Predictions by Bandwidth"))

####-----####
# anisotropic variogram/kernel
####-----####

scale_x <- 0.2
scale_y <- 1
rotations <- c(180, 22.5, 45, 67.5, 90, 112.5, 135, 157.5)
results_kernels <- data.frame(X=data$X, Y=data$Y)
results_kernels_coefs <- data.frame()
results_kernels_aic <- data.frame()
results_kernels_moran <- data.frame()

for (angle in rotations){

  rot_mtx <- matrix(c(cos(angle*(pi/180)), sin(angle*(pi/180)),
                     -sin(angle*(pi/180)), cos(angle*(pi/180))),
                   nrow=2)

  rotated_coords <- coords %*% rot_mtx
  rotated_coords <- as.matrix(coordinates(rotated_coords))

  scaled_coords <- as.matrix(data.frame(
    X = rotated_coords[,1] * scale_x,
    Y = rotated_coords[,2] * scale_y
  ))

  W <- dnearneigh(scaled_coords, 0, 1) # 300 distance threshold neighborhood
  W <- nb2listw(W)

  slm_model <- lagsarlm(formula,
                       data=data,
                       listw=W)
  y_prob <- ifelse(slm_model$fitted.values > 1, 1, slm_model$fitted.values)
  y_prob <- ifelse(y_prob < 0, 0, y_prob)
  y_prob <- plogis(log_link(y_prob))
  y_pred <- ifelse(y_prob > 0.5, 1, 0)
  # save fitted vals
  results_kernels[, paste0("angle_", angle)] <- y_pred

```

```

# save coefficient, AIC, and morans I results
results_kernels_coefs <- rbind(results_kernels_coefs, coef(summary(slm_model)))
results_kernels_aic <- rbind(results_kernels_aic,
                             data.frame(angle=angle[1],
                                           aic=slm_model[["AIC_lm.model"]]))
moran_i <- moran.test(residuals(slm_model), listw=W)
results_kernels_moran <- rbind(results_kernels_moran,
                                data.frame(angle = angle[1],
                                            Moran_I = moran_i[["estimate"]][["Moran I
                                            Exp = moran_i[["estimate"]][["Expectation
                                            Var = moran_i[["estimate"]][["Expectation

}

par(mfrow = c(1, 1))
burned_colors <- c('cadetblue2', 'firebrick2')
plot_raw <- ggplot(data, aes(x=X, y=Y, fill=as.factor(class))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = 'Observed Data', x='X', y='Y', fill='')

plot_baseline <- ggplot(results_baseline, aes(x=X, y=Y, fill=as.factor(round(fitted)))
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = 'Baseline Prediction', x='X', y='Y', fill='')

plot_x00 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_180'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = 'No Orientation Shift', x='X', y='Y', fill='')

plot_x0 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_22.5'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '22.5 Orientation', x='X', y='Y', fill='')

plot_x1 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_45'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '45 Orientation', x='X', y='Y', fill='')

plot_x2 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_67.5'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '67.5 Orientation', x='X', y='Y', fill='')

```

```

plot_x3 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_90'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '90 Orientation', x='X', y='Y', fill='')

plot_x4 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_112.5'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '112.5 Orientation', x='X', y='Y', fill='')

plot_x5 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_135'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '135 Orientation', x='X', y='Y', fill='')

plot_x6 <- ggplot(results_kernels, aes(x=X, y=Y, fill=as.factor('angle_157.5'))) +
  geom_tile() +
  scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
  labs(title = '157.5 Orientation', x='X', y='Y', fill='')

grid.arrange(plot_raw, plot_baseline, plot_x00, plot_x0,
             plot_x1, plot_x2, plot_x3,
             plot_x4, plot_x5, plot_x6,
             ncol=2, nrow=5, top=textGrob("Ship Data - Anisotropic Weighting SLM Pred

####-----####
# eigenvector spatial filtering
####-----####
# https://rdrr.io/cran/spmoran/man/esf.html
meig_f<- meigen_f(coords=coords)

# vif 3 cutoff
esfD <- esf(y=data$class, x=data[3:5],
           meig=meig_f,
           vif=3)
transformed_data <- data.frame(esfD$pred, data)
logit_esf <- glm(class ~ esfD.pred,
                data = transformed_data,
                family = binomial)

results_esf <- data.frame(X=data$X, Y=data$Y, fitted=logit_esf$fitted.values)

par(mfrow = c(1, 1))
ggplot(results_esf, aes(x=X, y=Y, fill=as.factor(round(fitted)))) +
  geom_tile() +

```

```

scale_fill_manual(values=burned_colors, labels=c('Absent', 'Present')) +
labs(title = 'Eigenvector Spatial Filter GLM Predictions', x='Easting', y='Northing')

####-----####
# empirical variogram plots
####-----####

class <- SpatialPointsDataFrame(coords=coords,
                                data=data.frame(value=data$class))
emp_variogram <- variogram(value ~ 1, data=class, width = 1)
class_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "Container Ship Presence Variogram",
        x = "Distance (d)", y = "Semivariance")

red <- SpatialPointsDataFrame(coords=coords,
                              data=data.frame(value=data$red))
emp_variogram <- variogram(value ~ 1, data =red, width = 1)
red_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "Red Channel Variogram",
        x = "Distance (d)", y = "Semivariance")

green <- SpatialPointsDataFrame(coords=coords,
                                data=data.frame(value=data$green))
emp_variogram <- variogram(value ~ 1, data =green, width = 1)
green_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "Green Channel Variogram",
        x = "Distance (d)", y = "Semivariance")

blue <- SpatialPointsDataFrame(coords=coords,
                              data=data.frame(value=data$red))
emp_variogram <- variogram(value ~ 1, data = blue, width = 1)
blue_vgm_plot <- ggplot(data=as.data.frame(emp_variogram)) +
  geom_point(aes(x=dist, y=gamma), col="blue", shape = 19) +
  labs(title = "Blue Channel Variogram",
        x = "Distance (d)", y = "Semivariance")

grid.arrange(class_vgm_plot,
              red_vgm_plot,
              green_vgm_plot,
              blue_vgm_plot,
              ncol=2, nrow=2,
              top=textGrob("Empirical Semivariograms"))

```

```

####-----####
# model performance comparison
####-----####

# prediction accuracy
# baseline
mean(data$class == round(results_baseline$fitted))
# bandwidths
mean(data$class == results_bandwidths$bandwidth_1)
mean(data$class == results_bandwidths$bandwidth_2)
mean(data$class == results_bandwidths$bandwidth_3)
mean(data$class == results_bandwidths$bandwidth_4)
# anisotropic kernel accuary
mean(data$class == results_kernels$angle_180)
mean(data$class == results_kernels$angle_22.5)
mean(data$class == results_kernels$angle_45)
mean(data$class == results_kernels$angle_67.5)
mean(data$class == results_kernels$angle_90)
mean(data$class == results_kernels$angle_112.5)
mean(data$class == results_kernels$angle_135)
mean(data$class == results_kernels$angle_157.5)
# esf
mean(data$class == round(results_esf$fitted))

#f1 scores
#baseline
precision <- posPredValue(as.factor(round(results_baseline$fitted)),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(round(results_baseline$fitted)),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

#bands
precision <- posPredValue(as.factor(results_bandwidths$bandwidth_1),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_1),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_2),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_2),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_3),

```

```

                                as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_3),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_bandwidths$bandwidth_4),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_bandwidths$bandwidth_4),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

# anisotropic
precision <- posPredValue(as.factor(results_kernels$angle_180),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_180),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_22.5),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_22.5),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_45),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_45),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_67.5),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_67.5),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_90),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_90),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_112.5),
                        as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_112.5),
                    as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

```

```

precision <- posPredValue(as.factor(results_kernels$angle_135),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_135),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

precision <- posPredValue(as.factor(results_kernels$angle_157.5),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(results_kernels$angle_157.5),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

# esf
precision <- posPredValue(as.factor(round(results_esf$fitted)),
                          as.factor(data$class), positive="1")
recall <- sensitivity(as.factor(round(results_esf$fitted)),
                     as.factor(data$class), positive="1")
(2 * precision * recall) / (precision + recall)

# morans I
moran.test(residuals(logit_baseline), listw=W)
round(results_bw_moran$Moran_I, 3)
round(results_kernels_moran$Moran_I, 3)
moran.test(residuals(logit_esf), listw=W)

```