Local likelihood estimation for covariance functions with spatially-varying parameters: the convoSPAT package for R

Mark D. Risser [a] and Catherine A. Calder [a *]

February 22, 2015

[a] Department of Statistics, The Ohio State University, OH 43210, USA

[*] Correspondance: Department of Statistics, 1958 Neil Avenue, Columbus, OH 43210. Email: calder@stat.osu.edu

Abstract

In spite of the interest in and appeal of convolution-based approaches for nonstationary spatial modeling, off-the-shelf software for model fitting does not as of yet exist. Convolution-based models are highly flexible yet notoriously difficult to fit, even with relatively small data sets. The general lack of pre-packaged options for model fitting makes it difficult to compare new methodology in nonstationary modeling with other existing nonstationary methods, and as a result most new models are simply compared to stationary models. Using a convolution-based approach, we present a new nonstationary covariance function for spatial Gaussian process models that allows for efficient computing in two ways: first, by representing the spatially-varying parameters via a discrete mixture or "mixture component" model, and second, by estimating the mixture component parameters through a local likelihood approach. In order to make computation for a convolution-based nonstationary spatial model readily available, this paper also presents and describes the convoSPAT package for R. The nonstationary model is fit to both a synthetic data set and a real data application involving annual precipitation to demonstrate the capabilities of the package.

1 Introduction

The Gaussian process is extremely popular modeling approach in modern-day spatial and environmental statistics, due largely to the fact that the second-order properties of the process can be completely defined through the covariance function. A broad literature on covariance function modeling exists, but traditional approaches are mostly based on assumptions of isotropy or stationarity, in which the covariance between the spatial process at two locations is a function of only the separation distance or separation vector, respectively. This modeling assumption is made mostly for convenience, and is almost never a realistic assumption in practice. As a result, a wide variety nonstationary covariance function models for Gaussian process models have been developed (e.g., Sampson and Guttorp, 1992, Higdon, 1998, Damian et al., 2001, Fuentes, 2001, Schmidt and O'Hagan, 2003, Paciorek and Schervish, 2006, Calder, 2008, Schmidt et al., 2011, Reich et al., 2011, and Vianna Neto et al., 2014), in which the spatial dependence structure is allowed to vary over the spatial region of interest. However, while these nonstationary approaches more appropriately model the covariance in the spatial process, most are also highly complex and require intricate model-fitting algorithms, making it very difficult to replicate their results in a general setting. Therefore, when new nonstationary methods are developed, their performance is usually compared to stationary models, for which a wide variety of software is available. In order to more accurately evaluate new nonstationary methods, pre-packaged and efficient options for fitting existing nonstationary models must be made available.

To address this need, we present a simplified version of the nonstationary spatial Gaussian process model introduced by Paciorek and Schervish (2006) in which the locally-varying geometric anisotropies are modeled using a "mixture component" approach, similar to the discrete mixture kernel convolution approach in Higdon (1998), while also allowing the underlying correlation structure to be specified by the modeler. The model can be extended to allow other properties to vary over space as well, such as the process variance and nugget effect. An additional degree of efficiency is gained by using local likelihood techniques to estimate the spatially-varying features of the spatial process; then, the locally estimated features are smoothed over space, similar to the approach of Fuentes (2002).

This paper also presents and describes the `convoSPAT` package for `R` for conducting a full analysis of spatial data using a nonstationary spatial Gaussian process model. The primary contribution of the package is to offer efficient model-fitting for nonstationary, point-referenced spatial data, even when the size of the data is relatively large (on the order of $n = 1000$ to 1500). The package is able to handle both a single realization of the spatial process as well as replicates. Finally, the paper demonstrates how the package can be used, and provides an analysis of both simulated and real data sets.

2 A convolution-based nonstationary covariance function

Process convolutions or moving average models are popular constructive methods for specifying a nonstationary process model. In general, a spatial stochastic process $Y(\cdot)$ on $G \subset \mathcal{R}^d$ can be defined by the kernel convolution

$$Y(\mathbf{s}) = \int_{\mathcal{R}^d} K_{\mathbf{s}}(\mathbf{u}) dW(\mathbf{u}), \tag{1}$$

where $W(\cdot)$ is a d-dimensional stochastic process and $K_{\mathbf{s}}(\cdot)$ is a (possibly parametric) spatially-varying kernel function centered at s $\in G$. Higdon (2002) summarizes the extremely flexible class of spatial process models defined by (1): see, for example, Barry and Ver Hoef (1996), Ver Hoef et al. (2004), Ickstadt and Wolpert (1998), Higdon (1998), Ver Hoef et al. (2004), and Hoef and Barry (1998).

The kernel convolution (1) defines a mean-zero nonstationary spatial Gaussian process (GP) if $W(\cdot)$ is chosen to be d-dimensional Brownian motion. An additional benefit of using (1) is that in this case the kernel functions completely specify the second-order properties of the GP through the covariance function

$$Cov(Y(\mathbf{s}), Y(\mathbf{s}')) = E\big[Y(\mathbf{s})Y(\mathbf{s}')\big] = \int_G K_{\mathbf{s}}(\mathbf{u}) K_{\mathbf{s}'}(\mathbf{u}) d\mathbf{u}, \tag{2}$$

where $\mathbf{s}, \mathbf{s}' \in G$. The popularity of this approach is due largely to the fact that it is much easier to specify kernel functions than a covariance function directly, since the kernel functions only require

$$\int_{\mathcal{R}^d} K_{\mathbf{s}}(\mathbf{u}) d\mathbf{u} < \infty$$

and

$$\int_{\mathcal{R}^d} K^2_{\mathbf{s}}(\mathbf{u}) d\mathbf{u} < \infty.$$

A famous result (Thiébaux, 1976; Thiébaux and Pedder, 1987) uses a parametric class of kernel functions in (2) to give a closed-form covariance function; this result was later extended (Paciorek, 2003; Paciorek and Schervish, 2006; Stein, 2005) to show that

$$C^{NS}(\mathbf{s}, \mathbf{s}'; \boldsymbol{\theta}) = \sigma(\mathbf{s})\sigma(\mathbf{s}') \frac{|\boldsymbol{\Sigma}(\mathbf{s})|^{1/4} |\boldsymbol{\Sigma}(\mathbf{s}')|^{1/4}}{\left| \frac{\boldsymbol{\Sigma}(\mathbf{s}) + \boldsymbol{\Sigma}(\mathbf{s}')}{2} \right|^{1/2}} g\left(\sqrt{Q(\mathbf{s}, \mathbf{s}')} \right), \tag{3}$$

is a valid, nonstationary, parametric covariance function for $\mathcal{R}^d, d \geq 1$, when $g(\cdot)$ is chosen to be a valid correlation function for $\mathcal{R}^d, d \geq 1$. In (3), $\boldsymbol{\theta}$ is a generic parameter vector, $\sigma(\cdot)$ represents a spatially-varying standard deviation, $\boldsymbol{\Sigma}(\cdot)$ is a $d \times d$ matrix that represents spatially-varying local anisotropy (controlling both the range and direction of dependence), and

$$Q(\mathbf{s}, \mathbf{s}') = (\mathbf{s} - \mathbf{s}')^T \left(\frac{\boldsymbol{\Sigma}(\mathbf{s}) + \boldsymbol{\Sigma}(\mathbf{s}')}{2} \right)^{-1} (\mathbf{s} - \mathbf{s}') \tag{4}$$

is a scaled distance. Furthermore, choosing $g(\cdot)$ to be the Matérn correlation function also allows for the introduction of $\nu(\cdot)$, a spatially-varying smoothness (Stein, 2005). While using (3) no longer requires the notion of kernel convolution, we refer to $\boldsymbol{\Sigma}(\cdot)$ as the kernel matrix, since it was originally defined as the covariance matrix of a Gaussian kernel function (Thiébaux, 1976; Thiébaux and Pedder, 1987). The covariance function (3) is extremely flexible, and has been used in various forms throughout the literature, e.g., Paciorek and Schervish (2006), Anderes and Stein (2011), Kleiber and Nychka (2012), Katzfuss (2013), and Risser and Calder (2014).

3 A nonstationary spatial Gaussian process model

The Gaussian process model defined by the covariance function (3) can be expanded to include a non-zero mean and measurement error for a full spatial model as follows. Define $\{Z(\mathbf{s}), \mathbf{s} \in G\}$, $G \subset \mathcal{R}^d$, to be the mean-corrected observed value of $Y(\mathbf{s})$, a latent nonstationary spatial process. Then, we can model

$$Z(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T\boldsymbol{\beta} + Y(\mathbf{s}) + \epsilon(\mathbf{s}) \tag{5}$$

where $E[Z(\mathbf{s})] = \mathbf{x}(\mathbf{s})^T\boldsymbol{\beta}$, $\mathbf{x}(\mathbf{s})$ is a p-vector of observable covariates for location $\mathbf{s}$, $\boldsymbol{\beta} \in \mathcal{R}^p$ are mean coefficients (note, however, that a linear mean function need not be assumed), the $\epsilon(\mathbf{s})$ are conditionally independent $\mathcal{N}(0, \tau^2(\mathbf{s}))$ (given $\tau^2(\cdot)$), and, conditional on parameters, $Y(\cdot) \sim \mathcal{GP}(\mathbf{0}, C^{NS})$; $\epsilon(\cdot)$ and $Y(\cdot)$ are independent. (Note: $\mathcal{N}_q(a, B)$ denotes the q-variate Gaussian distribution with mean vector a and covariance B.) Furthermore, suppose we have observations which are a partial realization of this random process, taken at a fixed, finite set of n spatial locations $\{\mathbf{s}_1, ..., \mathbf{s}_n\} \in G$, giving the random (observed) vector $\mathbf{Z} = (Z(\mathbf{s}_1), ..., Z(\mathbf{s}_n))'$, which, following (5), has a multivariate Gaussian distribution, conditional on the unobserved latent process and all other model parameters. Integrating out the process $\mathbf{Y}$ from (5), we obtain the marginal likelihood of the observed data $\mathbf{Z}$ given all parameters, which is $\mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \mathbf{D} + \boldsymbol{\Omega})$, where $\mathbf{X} = \big(\mathbf{x}(\mathbf{s}_1)^T, \dots, \mathbf{x}(\mathbf{s}_n)^T\big)^T$, $\boldsymbol{\Omega}$ has elements $\Omega_{ij} = C^{NS}(\mathbf{s}_i, \mathbf{s}_j; \boldsymbol{\theta})$, and $\mathbf{D} = diag\big[\tau^2(\mathbf{s}_1), \dots, \tau^2(\mathbf{s}_n)\big]$. For a particular application, the practitioner can specify the underlying correlation structure (through $g(\cdot)$) as well as determine which of $\{\boldsymbol{\Sigma}(\cdot), \sigma(\cdot), \tau^2(\cdot)\}$ (or $\nu(\cdot)$, if the Matérn is used) should be fixed or allowed to vary spatially. However, some care should be taken in choosing which quantities should be spatially-varying: for example, Anderes and Stein (2011) note that the allowing both $\boldsymbol{\Sigma}(\cdot)$ and $\nu(\cdot)$ to vary over space leads to issues with identifiability.

3.1 Discrete mixture representation

One way to reduce the computational demands of a spatial model which uses the covariance function (3) is by characterizing the nonstationary behavior of a spatial process through the discretized basis function approach of Higdon (1998). In the original paper, Higdon (1998) estimated the Gaussian kernel function for a generic location to be a weighted average of "basis" kernel functions, estimated locally over the spatial region of interest. However, since the use of Gaussian kernel functions results in undesirable smoothness properties (see, e.g., Paciorek and Schervish, 2006), we instead use a related "mixture component" approach, in which the parametric quantities for an arbitrary spatial location are defined as a mixture of spatially-varying parameter values associated with a fixed set of component locations. Specifically, in this new approach, define mixture component locations $\{\mathbf{b}_k : k = 1, \dots, K\}$ with corresponding parameters $\{(\boldsymbol{\Sigma}_k, \sigma^2_k, \tau^2_k, \nu_k) : k = 1, \dots, K\}$ (which are the kernel matrix, variance, nugget variance, and smoothness, respectively). Then, for $\phi \in \{\boldsymbol{\Sigma}, \sigma^2, \tau^2, \nu\}$, the parameter set for an arbitrary location $\mathbf{s} \in G$ is calculated as

$$\phi(\mathbf{s}) = \sum_{k=1}^{K} w_k(\mathbf{s})\phi_k, \tag{6}$$

where

$$w_k(\mathbf{s}) \propto \exp\left\{-\frac{||\mathbf{s} - \mathbf{b}_k||^2}{2\lambda_w}\right\} \tag{7}$$

such that $\sum_{k=1}^{K} w_k(\mathbf{s}) = 1$. For example, the kernel matrix for $\mathbf{s} \in G$ is $\boldsymbol{\Sigma}(\mathbf{s}) = \sum_{k=1}^{K} w_k(\mathbf{s})\boldsymbol{\Sigma}_k$. In (7), λ_w acts as a tuning parameter, ensuring that the rate of decay in the weighting function is appropriate for both the data set and scale of the spatial domain. Using this approach, the number of parameters is now linear in K, the number of mixture component locations, instead of n, the sample size. Furthermore, this specification still enables the modeler to choose which parameters should be spatially-varying: the kernel matrices, the process variance, the nugget variance, and the smoothness.

3.2 Prediction

Conditional on parameter estimates, the kriging predictor (i.e., the best linear unbiased predictor, REF) and standard errors are calculated as follows. Again, taking $\boldsymbol{\theta}$ to be a generic parameter vector including all variance and covariance parameters, define $\mathbf{Z}^* = (Z(\mathbf{s}_1^*), ..., Z(\mathbf{s}_m^*))'$ to be a vector of the process values at all prediction locations of interest. The Gaussian process model (5) implies that

$$\left[\begin{array}{c|c} \mathbf{Z} & \multirow{2}{*}{$\boldsymbol{\beta}, \boldsymbol{\theta}$} \\ \mathbf{Z}^* & \end{array}\right] \sim \mathcal{N}_{n+m}\left(\begin{bmatrix} \mathbf{X}\boldsymbol{\beta} \\ \mathbf{X}^*\boldsymbol{\beta} \end{bmatrix}, \begin{bmatrix} \mathbf{D}+\boldsymbol{\Omega} & \boldsymbol{\Omega}_{\mathbf{Z}\mathbf{Z}^*} \\ \boldsymbol{\Omega}_{\mathbf{Z}^*\mathbf{Z}} & \mathbf{D}^*+\boldsymbol{\Omega}^* \end{bmatrix}\right),$$

where $Cov(\mathbf{Z}^*) = \mathbf{D}^* + \boldsymbol{\Omega}^*$ and $Cov(\mathbf{Z}, \mathbf{Z}^*) = \boldsymbol{\Omega}_{\mathbf{Z}\mathbf{Z}^*}$. By the properties of the multivariate Gaussian distribution,

$$\mathbf{Z}^*|\mathbf{Z} = \mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\theta} \sim \mathcal{N}_m(\boldsymbol{\mu}_{\mathbf{Z}^*|\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{Z}^*|\mathbf{z}}), \tag{8}$$

where

$$\boldsymbol{\mu}_{\mathbf{Z}^*|\mathbf{z}} = \mathbf{X}^*\boldsymbol{\beta} + \boldsymbol{\Omega}_{\mathbf{Z}^*\mathbf{Z}}(\mathbf{D}+\boldsymbol{\Omega})^{-1}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}), \tag{9}$$

and

$$\boldsymbol{\Sigma}_{\mathbf{Z}^*|\mathbf{z}} = (\mathbf{D}^* + \boldsymbol{\Omega}^*) - \boldsymbol{\Omega}_{\mathbf{Z}^*\mathbf{Z}}(\mathbf{D}+\boldsymbol{\Omega})^{-1}\boldsymbol{\Omega}_{\mathbf{Z}\mathbf{Z}^*}. \tag{10}$$

Using the plug-in estimates $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\theta}}$, the kriging predictor is then $\widehat{\boldsymbol{\mu}}_{\mathbf{Z}^*|\mathbf{z}}$ with corresponding kriging prediction errors as the square root of the diagonal elements of $\widehat{\boldsymbol{\Sigma}}_{\mathbf{Z}^*|\mathbf{z}}$.

3.2.1 Out-of-sample evaluation criteria

Two cross-validation evaluation criteria can be used to assess the fit of the nonstationary spatial model (5). First, the mean squared prediction error

$$MSPE = \frac{1}{m}\sum_{j=1}^{m}(z_j^* - \hat{z}_j^*)^2, \tag{11}$$

where z_j^* is the jth held-out observed value and $\hat{z}_j^*$ is the corresponding kriging predictor (from (9)). The MSPE is a point-wise measure of model fit, and smaller MSPE indicates better predictions.

Second, the continuous rank probability score will be used (a proper scoring rule; see Gneiting and Raftery, 2007). For the jth prediction, this is defined as

$$CRPS_j \equiv CRPS(F_j, z_j^*) = \int_{-\infty}^{\infty} \big(F_j(x) - 1\{x \geq z_j^*\}\big)^2 dx, \tag{12}$$

where $F_j(\cdot)$ is the cumulative distribution function for the predictive distribution of z_j^* given the training data and $1\{\cdot\}$ is the indicator function. In this case, given that the predictive CDF is Gaussian (conditional on parameters; see (8)), a computational shortcut can be used for calculating (12): when F is Gaussian with mean μ and variance σ^2,

$$CRPS\big(F, z_j^*\big) = \sigma\left[\frac{1}{\sqrt{\pi}} - 2\cdot\phi\left(\frac{z_j^* - \mu}{\sigma}\right) - \frac{z_j^* - \mu}{\sigma}\left(2\cdot\Phi\left(\frac{z_j^* - \mu}{\sigma}\right) - 1\right)\right],$$

where ϕ and Φ denote the probability density and cumulative distribution functions, respectively, of a standard Gaussian random variable. The reported metric will be the average over all holdout locations, $\widehat{CRPS} = m^{-1}\sum_{j=1}^m \widehat{CRPS}_j$. CRPS measures the fit of the predictive density; smaller CRPS indicates better model fit.

4 Computationally efficient inference

As discussed in Section 1, fast and efficient inference for a nonstationary process convolution model has yet to be made readily available for general use. In spite of its popularity, (3) always requires some kind of constraints and has suffered from a lack of widespread use due to the complexity of the requisite model fitting and limited pre-packaged options. Focusing on the spatially-varying local anisotropy matrices $\mathbf{\Sigma}(\cdot)$, the covariance function (3) requires a kernel matrix at every observation and prediction location of interest. Paciorek and Schervish (2006) accomplish this by modeling $\mathbf{\Sigma}(\cdot)$ as itself a (stationary) stochastic process, assigning Gaussian process priors to the elements of the spectral decomposition of $\boldsymbol{\Sigma}(\cdot)$; alternatively, Katzfuss (2013) uses a basis function representation of $\boldsymbol{\Sigma}(\cdot)$. Both of these models are highly parameterized and require intricate Markov chain Monte Carlo methods for model fitting. Even the approach of Risser and Calder (2014), which uses covariates and has a greatly reduced parameter space, requires a good deal of computational complexity.

The model that we propose provides efficiency in two ways: first, from the model itself, which uses a discrete mixture representation (see Section 3.1), and second, by fitting the mixture components of the model locally, using the idea of local likelihood estimation (Tibshirani and Hastie, 1987).

4.1 Local likelihood estimation

Using the discrete mixture representation of (6), a "full likelihood" approach to parameter estimation could be taken, in either a Bayesian or maximum likelihood framework, although the optimization in a maximum likelihood approach could become intractable for either moderately large K or large n. However, since the primary goal of this new methodology is computational speed, a further degree of efficiency can be gained by using local likelihood estimation (LLE), due to Tibshirani and Hastie (1987).

Before discussing the local likelihood approach, we outline a restricted maximum likelihood (REML; see Patterson and Thompson, 1971, Patterson and Thompson, 1974, and Kitanidis, 1983) approach for separating estimation of the mean parameters $\boldsymbol{\beta}$ from the covariance parameters $\boldsymbol{\theta}$. The full log-likelihood for $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ in (5) is

$$\mathcal{L}^F(\boldsymbol{\beta}, \boldsymbol{\theta}; \mathbf{Z}) = -\frac{1}{2}\log|\boldsymbol{\Omega} + \mathbf{D}| - \frac{1}{2}(\mathbf{Z} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{\Omega} + \mathbf{D})^{-1}(\mathbf{Z} - \mathbf{X}\boldsymbol{\beta}); \tag{13}$$

a standard maximum likelihood approach would set out to maximize $\mathcal{L}^F(\boldsymbol{\beta}, \boldsymbol{\theta}; \mathbf{Z})$ directly. REML, on the other hand, uses a (log) likelihood based on $n - p$ linearly independent linear combinations of the data that have an expected value of zero for all possible $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$. Regardless of which set of linearly independent combinations is chosen, the "restricted" log-likelihood, which depends only on $\boldsymbol{\theta}$, is

$$\mathcal{L}^R(\boldsymbol{\theta}; \mathbf{Z}) = -\frac{1}{2}\log|\boldsymbol{\Omega} + \mathbf{D}| - \frac{1}{2}\log|\mathbf{X}^T(\boldsymbol{\Omega} + \mathbf{D})^{-1}\mathbf{X}| - \frac{1}{2}\mathbf{Z}^T\mathbf{P}\mathbf{Z}, \tag{14}$$

where

$$\mathbf{P} = (\boldsymbol{\Omega} + \mathbf{D})^{-1} - (\boldsymbol{\Omega} + \mathbf{D})^{-1}\mathbf{X}\big(\mathbf{X}^T(\boldsymbol{\Omega} + \mathbf{D})^{-1}\mathbf{X}\big)^{-1}\mathbf{X}^T(\boldsymbol{\Omega} + \mathbf{D})^{-1}. \tag{15}$$

The REML estimate of $\boldsymbol{\theta}$ is obtained by maximizing $\mathcal{L}^R(\boldsymbol{\theta}; \mathbf{Z})$, and the estimate of $\boldsymbol{\beta}$ is the generalized least squares estimate

$$\widehat{\boldsymbol{\beta}} = \big(\mathbf{X}^T(\widehat{\boldsymbol{\Omega}} + \widehat{\mathbf{D}})^{-1}\mathbf{X}\big)^{-1}\mathbf{X}^T(\widehat{\boldsymbol{\Omega}} + \widehat{\mathbf{D}})^{-1}\mathbf{Z}, \tag{16}$$

which is obtained by plugging in $\widehat{\boldsymbol{\theta}}$ to calculate $\widehat{\boldsymbol{\Omega}}$ and $\widehat{\mathbf{D}}$. These parameter estimates can then be plugged in to $\widehat{\boldsymbol{\mu}}_{\mathbf{Z}^*|\mathbf{z}}$ and $\widehat{\boldsymbol{\Sigma}}_{\mathbf{Z}^*|\mathbf{z}}$ to obtain predictions and prediction standard errors.

In the LLE approach, instead of optimizating (14) directly we will set out to optimize $\mathcal{L}_k^R(\boldsymbol{\theta}_{N_k}; \mathbf{Z}_{N_k})$, where $N_k \equiv N_k(r)$ is a neighborhood for each mixture component location $\mathbf{b}_k$ that depends on the radius r, such that

$$N_k = \big\{\mathbf{s}_i \in \{\mathbf{s}_1, \ldots, \mathbf{s}_n\} : \{||\mathbf{s}_i - \mathbf{b}_k|| \leq r\}\big\}$$

and

$$\mathbf{Z}_{N_k} = \big\{Z(\mathbf{s}) : \mathbf{s} \in N_k\big\}.$$

Correspondingly, $\boldsymbol{\theta}_{N_k} = (\boldsymbol{\Sigma}_k, \sigma_k^2, \tau_k^2, \nu_k)$. The radius r defines the "span" (Tibshirani and Hastie, 1987) or window size for each mixture component. The restricted log-likelihood for neighborhood N_k will be based on a stationary version of the spatial model (5), namely

$$\widetilde{Z}(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T\widetilde{\boldsymbol{\beta}} + \widetilde{Y}(\mathbf{s}) + \widetilde{\epsilon}(\mathbf{s}), \tag{17}$$

where $\widetilde{Y}(\cdot)$ is a stationary, mean-zero spatial process with covariance function

$$C^S(\mathbf{s}-\mathbf{s}') = \sigma^2 g\left(||\boldsymbol{\Sigma}^{-1/2}(\mathbf{s}-\mathbf{s}')||\right),$$

the $\widetilde{\epsilon}(\cdot)$ are independent and identically distributed as $\mathcal{N}(0,\tau^2)$, conditional on τ^2, and again $\widetilde{Y}(\cdot)$ and $\widetilde{\epsilon}(\cdot)$ are independent. Again, in a REML framework, only the variance and covariance parameters $\{\boldsymbol{\Sigma}_k, \sigma_k^2, \tau_k^2, \nu_k\}$ need to be estimated for each $k = 1,\ldots,K$. No estimates will be obtained for the local mean coefficient vector $\widetilde{\boldsymbol{\beta}}$, as all of the mean parameters will be estimated globally.

One final note regarding the estimation of the kernel matrices: the kernel matrix for mixture component location k will be obtained through estimating the parameters of its spectral decomposition, namely λ_1, λ_2, and η, where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \cos(\eta) & -\sin(\eta) \\ \sin(\eta) & \cos(\eta) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos(\eta) & \sin(\eta) \\ -\sin(\eta) & \cos(\eta) \end{bmatrix}. \tag{18}$$

Here, λ_1 and λ_2 are eigenvalues and represent squared ranges (such that $\lambda_1 > 0$ and $\lambda_2 > 0$) and η represents an angle of rotation, constrained to lie between 0 and $\pi/2$ for identifiability purposes (Katzfuss, 2013).

Returning to the full dataset, the model (5) can be fit by plugging REML estimates $\{\widehat{\boldsymbol{\Sigma}}_k, \widehat{\sigma}_k^2, \widehat{\tau}_k^2, \widehat{\nu}_k : k = 1,\ldots,K\}$ into the covariance function (3) using the discrete basis representation (6) to calculate the likelihood for the observed data. Variance quantities that are not specified to be spatially-varying can then be estimated again using REML with the spatially-varying components considered fixed. For example, if for a particular model only $\boldsymbol{\Sigma}(\cdot)$ is allowed to vary spatially and the smoothness is fixed, it remains to estimate the overall nugget τ^2 and variance σ^2. The restricted Gaussian likelihood for these parameters is then

$$\mathcal{L}^R(\sigma^2,\tau^2;\mathbf{Z},\mathbf{R}) = -\frac{1}{2}\log\left|\sigma^2\mathbf{R}+\tau^2\mathbf{I}_n\right| - \frac{1}{2}\log\left|\mathbf{X}^T(\sigma^2\mathbf{R}+\tau^2\mathbf{I}_n)^{-1}\mathbf{X}\right| - \frac{1}{2}\mathbf{Z}^T\mathbf{P}\mathbf{Z},$$

where $\mathbf{R}$ is the correlation matrix, i.e., the matrix calculated using (3) without the $\sigma(\cdot)$ terms, and $\mathbf{P}$ is defined as in (15). Once all of the covariance parameters have been estimated, the estimate of $\boldsymbol{\beta}$ can be calculated as in (16).

Using this model requires both the number and placement of mixture component locations $\{\mathbf{b}_k : k = 1, \ldots, K\}$, selecting which of the spatial dependence parameters should be fixed or allowed to vary spatially, the tuning parameter for the weighting function λ_w, and the fitting radius r. Parameter estimates for this model are likely to be sensitive to the choice of K and the placement of mixture component locations. Furthermore, Tibshirani and Hastie (1987) discuss the importance of choosing r, which specifies the "span size," suggesting that the model should be fit using a range of r values, and use a global criterion such as the maximized overall likelihood, cross-validation, or Akaike's Information Criterion to choose the final model. This strategy could either be implemented on a trial-and-error basis or in an automated scheme. Of course, regardless of the number and locations of the mixture component centroids, the radius r should be chosen such that a sufficiently large enough number of data points are used to estimate a local stationary model.

While different in both motivation and nature, the model outlined above is related to the local likelihood method described in Anderes and Stein (2011), which ties together locally stationary models to estimate a globally nonstationary model. The model in Anderes and Stein (2011) involves optimizing a sum of weighted increments of local log-likelihoods, where the weights are estimated smoothly using a smoothing kernels. Alternatively, our approach estimates spatially-varying parameters locally using only a subset of the data, then fixing the global parameters according to (6). Both of these approaches avoid the lack-of-smoothness issues innate to other similar segmentation approaches, such as Fuentes (2001) or the *ad hoc* nonstationary kriging approach in Paciorek and Schervish (2006), which Anderes and Stein (2011) call "hard thresholding" local likelihood estimates. Like Anderes and Stein (2011), our approach avoids the problem of non-smooth local parameter estimates implicit to hard thresholding methods by using the mixture component representation.

5 Using the `convoSPAT` package for `R`

5.1 Nonstationary model fitting

The primary components of the `convoSPAT` package are the `NSconvo.fit` and `NSconvo.pred` functions which fit the nonstationary model (5) and provide predictions, respectively. Much of the underlying coding of this package relies on base functions of the `geoR` (Ribeiro Jr. and Diggle, 2001) package, and therefore uses similar data structures.

The `NSconvo.fit` function takes the following arguments (with defaults as given):

```
NSconvo.fit( geodata, coords = geodata$coords, data = geodata$data,
             cov.model = "exponential", mean.model = data ~ 1,
             mc.locations = NULL, N.mc = NULL,
             mc.kernels = NULL, fit.radius = NULL, lambda.w = NULL,
             ns.nugget = FALSE, ns.variance = FALSE,
             local.pars.LB = NULL, local.pars.UB = NULL,
             global.pars.LB = NULL, global.pars.UB = NULL,
             local.ini.pars = NULL, global.ini.pars = NULL )
```

Required inputs are a `geodata` object (or, equivalently, separate specification of the `coords` with locations and `data` with the variable of interest), the number of mixture component locations (`N.mc`), and the fit radius (previously denoted r). The user may specify a covariance model from the `geoR` options `cauchy`, `matern`, `circular`, `cubic`, `gaussian`, `exponential`, `spherical`, or `wave`, as well as a mean model through the usual formula notation (a constant mean is the default). For most applications, the user will want to specify the mixture component locations: the default is to create an evenly spaced grid over the spatial domain of interest, which may not be appropriate if the spatial domain is non-rectangular. The tuning parameter for the weighting function λ_w is defined by `lambda.w`. The default for λ_w is fixed to be the square of one-half of the minimum distance between mixture component locations, or $(0.5\min\{||\mathbf{b}_k - \mathbf{b}_{k'}||\})^2$, but may also be specified by the user. The user may also specify if either the nugget variance or process variance is to be spatially-varying by setting either `ns.nugget = TRUE` or `ns.variance = TRUE` (or both). If the mixture component kernels themselves are pre-specified (e.g., based on expert opinion), these may also be passed into the function, which will greatly reduce computational time.

Note that if the data and coordinates are not specified as a `geodata` object, the `data` argument for this function can accommodate replicates. This might be of interest for applications similar to the ones in Sampson and Guttorp (1992), in which the replicates represent repeated observations over time that have been temporally detrended. In this case, the model will assume a constant spatial dependence structure as well as a constant mean function over the replicates.

The optimization method used within `optim()` for this package is `"L-BFGS-B"`, which allows for the specification of upper and lower bounds for each parameter with respect to the optimization search. The upper and lower bounds may be passed to the function via `local.pars.LB`, `local.pars.UB`, `global.pars.LB`, and `global.pars.UB`. The local limits require vectors of length five, with bounds for the local parameters $\lambda_1, \lambda_2, \tau^2$, σ^2, and ν, while the global limits require vectors of length three, with bounds for the global parameters τ^2, σ^2, and ν. Default values for these limits are as follows: for both the global and local parameter estimation, the lower bounds for $\lambda_1, \lambda_2, \sigma^2, \tau^2$, and ν are fixed at 1e-5; the upper bound for the smoothness ν will be fixed to 100. The upper bounds for the variance and kernel parameters, on the other hand, will be specific to the application: for the nugget variance (τ^2) and process variance (σ^2), the upper bound will be $4\widehat{\sigma}^2_{OLS}$ (where $\widehat{\sigma}^2_{OLS}$ is the error variance estimate from a standard ordinary least squares procedure); the upper bound for λ_1 and λ_2 will be half of the maximum interpoint distance between observation locations in the data set. The bounds for η are fixed at 0 and $\pi/2$.

The final options in the `NSconvo.fit` function involve `local.ini.pars` and `global.ini.pars`, which specify the initial values used for the local and global calls of `optim()`, respectively. As with the limits, `local.ini.pars` is a vector of length five, with initial values for the local parameters $\lambda_1, \lambda_2, \tau^2$, σ^2, and ν, while `global.ini.pars` is a vector of length three, with initial values for the global parameters τ^2, σ^2, and ν. The default for these inputs are as follows: $\lambda_{1,init} = \lambda_{2,init} = c/10$, where c is the maximum interpoint distance between observation locations, $\tau^2_{init} = 0.1\widehat{\sigma}^2_{OLS}$, $\sigma^2_{init} = 0.9\widehat{\sigma}^2_{OLS}$, and $\nu_{init} = 1$.

When the `NSconvo.fit` function is called, the progress of the model fitting will be printed in real time. As the function fits the locally stationary models for each mixture component location, a line of text will print with information counting the mixture component location and number of observations that are currently being used for estimation. After the local models have been fit for each mixture component location, a line of text will print notifying the user that the variance parameters are being estimated globally.

A function which may be helpful before running `NSconvo.fit` is the `mc.N` function, which returns the number of observations which will be used to fit each local model for a particular set of mixture component locations and fit radius. The inputs to the function are

```
mc.N( coords, mc.locations, fit.radius )
```

where `coords` are the observation locations for the full data set, `mc.locations` are the mixture component locations, and `fit.radius` is the fitting radius. Using this function can help the user ensure that enough locations will fall within each local fitting radius.

After the model fitting has completed, the resulting `NSconvo.fit` object can be passed to the `summary.NS` function to quickly summarize the fitted model. Among other things, a `NSconvo.fit` object includes:

`mc.kernels`, which contains the estimated kernel matrices for the mixture component locations,

`mc.locations`, which contains the mixture component locations,

`MLEs.save`, which includes a data frame of the locally-estimated stationary model parameters for each mixture component location,

`kernel.ellipses`, which includes the estimated kernel ellipse for each location in `coords`,

`beta.GLS`, the generalized least squares estimate of $\boldsymbol{\beta}$,

`beta.cov`, the estimated covariance matrix of $\widehat{\boldsymbol{\beta}}$,

`tausq.est`, the estimate of the nugget variance – either a constant (if estimated globally) or a vector with the estimated nugget variance for each location in `coords`,

`sigmasq.est`, the estimate of the process variance – either a constant (if estimated globally) or a vector with the estimated process variance for each location in `coords`, `kappa.MLE`, the global estimate of the smoothness (for `cauchy` or `matern`),

`Cov.mat` and `Cov.mat.inv`, the estimated covariance matrix for the data and its inverse (respectively), and

`Xmat`, the design matrix for the mean model.

The `NSconvo.fit` object can also be passed to the `NSconvo.pred` function, which calculates predictions and prediction standard errors (from (9) and (10)). The `NSconvo.pred` function takes the following arguments:

```
NSconvo.pred( NSconvo.fit.obj, pred.coords, pred.covariates = NULL )
```

The `NSconvo.fit.obj` object is the output of `NSconvo.fit`, `pred.coords` is a list of the prediction locations of interest, and `pred.covariates` is a matrix of covariates for the prediction locations (the intercept is added automatically). Calculating the predictions when the dimension of `pred.coords` is large is computationally expensive, and a progress meter prints while the machine is working. A `NSconvo.pred` object includes:

`pred.means`, which contains the kriging predictor for each prediction location, and

`pred.SDs`, which contains the corresponding prediction standard error.

The predictions and standard errors can be plotted manually or by using the `plot.preds` function (described in Section 5.3).

5.2 Anisotropic model fitting

For the sake of comparison, the functions `Aniso.fit` and `Aniso.pred` are also given, which fit the stationary (anisotropic) model (17) to the full dataset. This function relies heavily on the package `geoR`, specifically utilizing the `likfit` function.

The `Aniso.fit` function takes the following arguments (with defaults as given):

```
Aniso.fit( geodata, coords = geodata$coords, data = geodata$data,
           cov.model = "exponential", mean.model = data ~ 1,
           iso.model = FALSE )
```

The inputs to this function are identical to the inputs to the nonstationary model fitting function. One new option is available, namely `iso.model`, used to indicate if the stationary model should be isotropic (`TRUE`) or anisotropic (`FALSE`). However, because the `likfit` object is required for the predictions, the structure of the output is slightly different than the corresponding function for the nonstationary model. Among other things, an `Aniso.fit` object includes:

`MLEs.save`, which includes the globally-estimated stationary model parameters,

`beta.GLS`, the generalized least squares estimate of $\boldsymbol{\beta}$,

`aniso.pars`, the global estimate of the anisotropy parameters λ_1, λ_2, and η, which define the anisotropy matrix by (18),

`aniso.mat`, which gives the global estimate of $\boldsymbol{\Sigma}$ from (18) in matrix form,

`tausq.est`, the global estimate of the nugget variance,

`sigmasq.est`, the global estimate of the process variance,

`kappa.MLE`, the global estimate of the smoothness (for `cauchy` or `matern`), and

`likfit.obj`, a `likfit()` object (to be used in the predict function).

Once the anisotropic model has been fit and the output stored, the fitted model object can be passed to the `Aniso.pred` function, which (similar to the nonstationary predict function) calculates predictions and prediction standard errors as in (9) and (10). Since this function uses `geoR` to do the heavy lifting in calculating the predictions, the arguments are slightly different:

```
Aniso.pred( Aniso.fit.obj, pred.coords, mean.model.d = ~ 1,
            mean.model.l = ~ 1 )
```

The `Aniso.fit.obj` object is the output of `Aniso.fit` and `pred.coords` is a list of the prediction locations of interest. Following `geoR` syntax, `mean.model.d` is a formula object which gives the mean formula with variables corresponding to the observed data, while `mean.model.l` is a formula object which gives the mean formula for the variables corresponding to the prediction locations. Similar to the nonstationary predict function, an `Aniso.pred` object includes:

`pred.means`, which contains the kriging predictor for each prediction location, and `pred.SDs`, which contains the corresponding prediction standard error.

5.3 Evaluation criteria and plotting functions

This package includes a function to quickly calculate the evaluation criteria, as well as functions to plot visualizations of the various components of the nonstationary model.

First, the `evaluate.CV` function calculates the MSPE, from (11), and the CRPS, from (12). The function inputs are simply

```
evaluate.CV( holdout.data, pred.mean, pred.SDs )
```

where `holdout.data` is the true held-out data and `pred.mean` and `pred.SDs` are output from one of the fit functions. Note that the user must perform the subsetting of the data. The output of `evaluate.CV` is simply the MSPE and CRPS, averaged over all hold-out locations.

Next, four plotting functions are provided to help visualize the output of either the stationary or nonstationary model. The first is `plot.mc`, which plots the estimated mixture component kernel ellipses for each mixture component location, while having the option to also show the fitting region for each mixture component location and the corresponding stationary anisotropy ellipse. The function is

```
plot.mc( mc.locations, mc.kernels, fit.radius, aniso.mat = NULL )
```

where `mc.locations` contains the mixture component locations, `mc.kernels` contains the locally estimated mixture component ellipses for each mixture component location, `fit.radius` is the fit radius, and an optional input is `aniso.mat`, which contains the stationary anisotropy ellipse. A similar plot can be made for the estimated ellipses corresponding to the observed locations, in order to more clearly show how the ellipses change smoothly over the spatial domain. This plot can be made by calling

```
plot.ellipses( locations, kernels, length )
```

where `locations` are the observed locations, `kernels` are the estimated kernel matrix for each location (obtained from the `NSconvo.fit` object), and `length` indicates how densely the kernel matrices should be plotted.

A simple function which plots the predictions and prediction standard errors is available in

```
plot.preds( locations, predmeans, predSDs, main1 = "Predictions",
            main2 = "Prediction standard errors",
            grid.locations = TRUE )
```

which uses the `image.plot` and `quilt.plot` functions from the `fields` package (Nychka et al., 2014). Again, `locations` is a list of the prediction locations, `predmeans` and `predSDs` contain the corresponding prediction means and standard deviations, `main1` and `main2` are optional plot titles, and `grid.locations` indicates if the prediction locations are on a grid (`TRUE`) or not (`FALSE`).

Finally, a function is provided to plot the estimated correlation between a specified reference location and the entire spatial domain of interest, using

```
plot.correlation( model.fit.obj, ref.loc, all.pred.locs,
                  NS.model = TRUE, grid = TRUE )
```

Here, `model.fit.obj` is an object from either `NSconvo.fit` or `Aniso.fit`, `ref.loc` is the reference location of interest, `all.pred.locs` is a list of all locations for which the correlation will be calculated, and `NS.model` is a logical input indicating if `model.fit.obj` is from `NSconvo.fit` (`TRUE`) or `Aniso.fit` (`FALSE`). The `grid` input specifies if the `all.pred.locs` locations lie on a rectangular grid (`TRUE`) or not (`FALSE`).

5.4 Other functions

Two additional functions are also provided to simulate data from the nonstationary model (5), and are used to create the simulated data set in Section 6. First is `f.mc.kernels`, which calculates the true mixture component kernel matrices through a generalized linear model framework for each of the elements of the kernel matrices' spectral decomposition as given in (18). The function, with default settings, is

```
f.mc.kernels( lat.min = 0, lat.max = 5, lon.min = 0, lon.max = 5,
              N.mc = 3^2, lam1.coef = c(-1.3, 0.5, -0.6),
              lam2.coef = c(-1.4, -0.1, 0.2),
              logit.eta.coef = c(0, -0.15, 0.15) )
```

The inputs `lat.mon`, `lat.max`, `lon.min`, and `lon.max` define a rectangular spatial domain, `N.mc` is the number of mixture component locations, and `lam1.coef`, `lam2.coef`, and `logit.eta.coef` define regression coefficients for the spatially-varying parameters λ_1, λ_2, and η. For example, taking `lam1.coef` $\equiv \boldsymbol{\beta}_{\lambda_1} = (\beta_0^{\lambda_1}, \beta_1^{\lambda_1}, \beta_2^{\lambda_1})^T$, the first eigenvector for an arbitrary location $\mathbf{s} = (s_1, s_2)$ is

$$\lambda_1(\mathbf{s}) = \exp\{\beta_0^{\lambda_1} + \beta_1^{\lambda_1} s_1 + \beta_2^{\lambda_1} s_2\}.$$

The other eigenvalue is calculated similarly. The angle of rotation, on the other hand, is calculated through the scaled inverse logit transformation

$$\eta(\mathbf{s}) = \frac{\pi}{2} \cdot \text{logit}^{-1}\big(\beta_0^{\eta} + \beta_1^{\eta} s_1 + \beta_2^{\eta} s_2\big),$$

where `logit.eta.coef` $\equiv \boldsymbol{\beta}_{\eta} = (\beta_0^{\eta}, \beta_1^{\eta}, \beta_2^{\eta})^T$. The default coefficients are those used to generate the simulated data set in Section 6, and were obtained by trial and error. The output of this function includes

`mc.locations`, which contains the mixture component locations, and

`mc.kernels`, which contains the mixture component kernels for each mixture component location.

The true mixture component kernel matrices generated in `f.mc.kernels` can be used to simulate a data set using the function

```
NSconvo.sim( grid = TRUE, lat.min = 0, lat.max = 5, lon.min = 0,
             lon.max = 5, N.obs = 20^2, sim.locations = NULL,
             mc.kernels.obj = NULL,
             mc.kernels = NULL, mc.locations = NULL,
             tausq = 0.1, sigmasq = 1, beta.coefs = 4, kappa = NULL,
             covariates = rep(1,N.obs), cov.model = "exponential" )
```

In this function, `grid` is a logical input specifying if the simulated data should lie on a grid (`TRUE`) or not (`FALSE`), `lat.mon`, `lat.max`, `lon.min`, and `lon.max` define the rectangular spatial domain, `N.obs` specifies the number of observed locations, `mc.kernels.obj` is an object from `f.mc.kernels`, `tausq`, `sigmasq`, `beta.coefs`, and `kappa` specify the true parameter values, `covariates` specifies the design matrix for the mean function, and `cov.model` specifies the covariance model. The output of this function is

`sim.locations`, which contains the simulated data locations,

`mc.locations`, which contains the mixture component locations,

`mc.kernels`, which contains the mixture component kernels for each mixture component location,

`kernel.ellipses`, which contains the kernel matrices for each simulated data location,

`Cov.mat`, which contains the true covariance matrix of the simulated data, and

`sim.data`, which contains the simulated data.

6 Example: simulated data

As a simple illustration, the nonstationary model (5) will be fit to an artificial data set simulated from the model. The data lie on a 25×25 grid (so that $n = 25^2 = 625$), and there are $K = 9$ mixture component locations with corresponding mixture component ellipses as given in Figure 1. Only the kernel matrices are allowed to vary spatially, an exponential correlation structure is used, and the mean structure contains the main effects of both longitude and latitude. The true parameter values are $\tau^2 = 0.1$, $\sigma^2 = 1$, $\beta_0 = 4$, $\beta_1 = -0.5$ (longitude coefficient), $\beta_2 = 0.5$ (latitude coefficient), and the fitting radius is set to $r = 2.3$ units. After much trial and error, the tuning parameter was fixed at $\lambda_w = 2$. A total of $m = 60$ of the simulated data points are used as a holdout sample. Figure 1 also provides the simulated data along with the holdout locations.

The simulated dataset can be loaded by running

```
> install.packages("convoSPAT")
> library(convoSPAT)
```

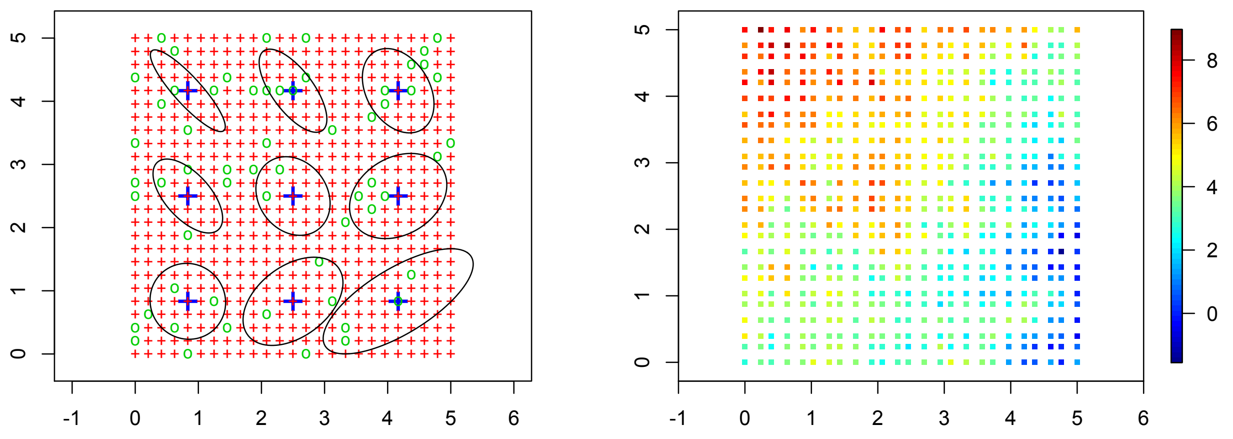

Figure 1: Left: true mixture component ellipses with observation locations (red) and holdout locations (green). Right: simulated data.

```
> load("simdata.RData")
```

The `simdata` object includes `sim.locations`, the simulated data locations, `mc.locations`, the mixture component locations, `mc.kernels`, the true mixture component kernel matrices, `sim.data`, the simulated data, and `holdout.index`, a vector of the 60 randomly sampled hold-out location indices. Figure 1 can be created with

```
> # Look at locations, ellipses, and data
> par(mfrow=c(1,2))
> plot(simdata$mc.locations, pch="+", asp=1, xlab="Longitude",
+      ylab="Latitude", xlim=c(-1,6), cex=2, col=4,
+      main="Mixture component locations/ellipses, observed (+) \n and
+             holdout (o) locations" )
> points(simdata$sim.locations[-holdout.index,], col=2, pch="+", cex = 0.5)
> points(simdata$sim.locations[holdout.index,], col=3, pch="o", cex = 0.5)
> for(i in 1:N.mc){
+   lines(ellipse(simdata$mc.kernels[,,i],
+      centre=simdata$mc.locations[i,], level=0.5))
+ }
> quilt.plot( simdata$sim.locations, simdata$sim.data, xlab="Longitude",
+  ylab="Latitude", main="Simulated data", asp=1, xlim=c(-1,6) )
```

Note that a replicated data set has also been included in the package, which contains ten replicates simulated using the same parameter values as the non-replicated data set (in fact, the first replicate of simulated data is identical to the data in the non-replicated data set). This can be loaded by

	True value	**OLS**	**Stationary model**	**Nonstationary model**
β_0	4	4.440	4.060	3.905
β_1	-0.5	-0.772	-0.698	-0.678
β_2	0.5	0.679	0.736	0.770
τ^2	0.1	–	0.090	0.107
σ^2	1	–	1.028	0.958
CRPS	–	–	0.502	0.469
MSPE	–	–	0.553	0.524
Computational time	–	–	0.44 minutes	6.92 minutes

Table 1: Parameter estimates from the simulated data, comparing the stationary and nonstationary models. [List details about the computing machine.]

running

```
> load("simdatareps.RData")
```

Other than the replicated data, the `simdatareps` data set contains all of the same objects as `simdata`.

The nonstationary model can be fitted to the non-hold-out data by

```
> NSfit.model <- NSconvo.fit(
+         coords = simdata$sim.locations[-simdata$holdout.index,],
+         data = simdata$sim.data[-simdata$holdout.index],
+         cov.model = "exponential",
+         fit.radius = 2.3, lambda.w = 2,
+         mc.locations = simdata$mc.locations,
+         mean.model = simdata$sim.data[-simdata$holdout.index]
+              ~ simdata$sim.locations[-simdata$holdout.index,1]
+                   + simdata$sim.locations[-simdata$holdout.index,2] )
```

Similarly, the anisotropic model can be fit to the non-hold-out data by

```
> anisofit.model <- Aniso.fit(
+         coords = simdata$sim.locations[-simdata$holdout.index,],
+         data = simdata$sim.data[-simdata$holdout.index],
+         cov.model = "exponential",
+         mean.model = simdata$sim.data[-simdata$holdout.index]
+               ~ simdata$sim.locations[-simdata$holdout.index,1]
+                   + simdata$sim.locations[-simdata$holdout.index,2] )
```

A summary of the results from each fitted model is provided in Table 1.

Predictions for the hold-out locations under each model can be calculated by calling

```
> pred.NS <- NSconvo.pred(
+    NSconvo.fit.obj = NSfit.model,
+    pred.coords = simdata$sim.locations[simdata$holdout.index,],
+    pred.covariates = simdata$sim.locations[simdata$holdout.index,] )
> pred.S <- Aniso.pred(
+    Aniso.fit.obj = anisofit.model,
+    pred.coords = simdata$sim.locations[simdata$holdout.index,],
+    mean.model.d = simdata$sim.data[-simdata$holdout.index]
+        ~ simdata$sim.locations[-simdata$holdout.index,1]
+        + simdata$sim.locations[-simdata$holdout.index,2],
+    mean.model.l = simdata$sim.data[simdata$holdout.index]
+         ~ simdata$sim.locations[simdata$holdout.index,1]
+         + simdata$sim.locations[simdata$holdout.index,2] )
```

after which the evaluation criteria can be calculated by

```
> evaluate.CV( holdout.data = simdata$sim.data[simdata$holdout.index],
+              pred.mean = pred.NS$pred.means, pred.SDs = pred.NS$pred.SDs )


> evaluate.CV( holdout.data = simdata$sim.data[simdata$holdout.index],
+              pred.mean = pred.S$pred.means, pred.SDs = pred.S$pred.SDs )
```

Calculating predictions on a finer resolution can be done as follows:

```
> grid.spacing <- 0.05
> grid.x <- seq( from = lon.min, to = lon.max, by = grid.spacing )
> grid.y <- seq( from = lat.min, to = lat.max, by = grid.spacing )
> grid.locations <- expand.grid( grid.x, grid.y )
> pred.locs <- matrix( c(grid.locations[,1], grid.locations[,2]),
+        ncol=2, byrow=F )
>
> pred.NS.all <- NSconvo.pred(
+        NSconvo.fit.obj = NSfit.model,
+        pred.coords = pred.locs,
+        pred.covariates = pred.locs )
> pred.S.all <- Aniso.pred(
+        Aniso.fit.obj = anisofit.model,
+        pred.coords = pred.locs,
+        mean.model.d = simdata$sim.data[-simdata$holdout.index]
+              ~ simdata$sim.locations[-simdata$holdout.index,1]
+                  + simdata$sim.locations[-simdata$holdout.index,2],
+        mean.model.l = ~ pred.locs[,1] + pred.locs[,2] )
```

“Filled-in” prediction maps with corresponding errors can be created by calling

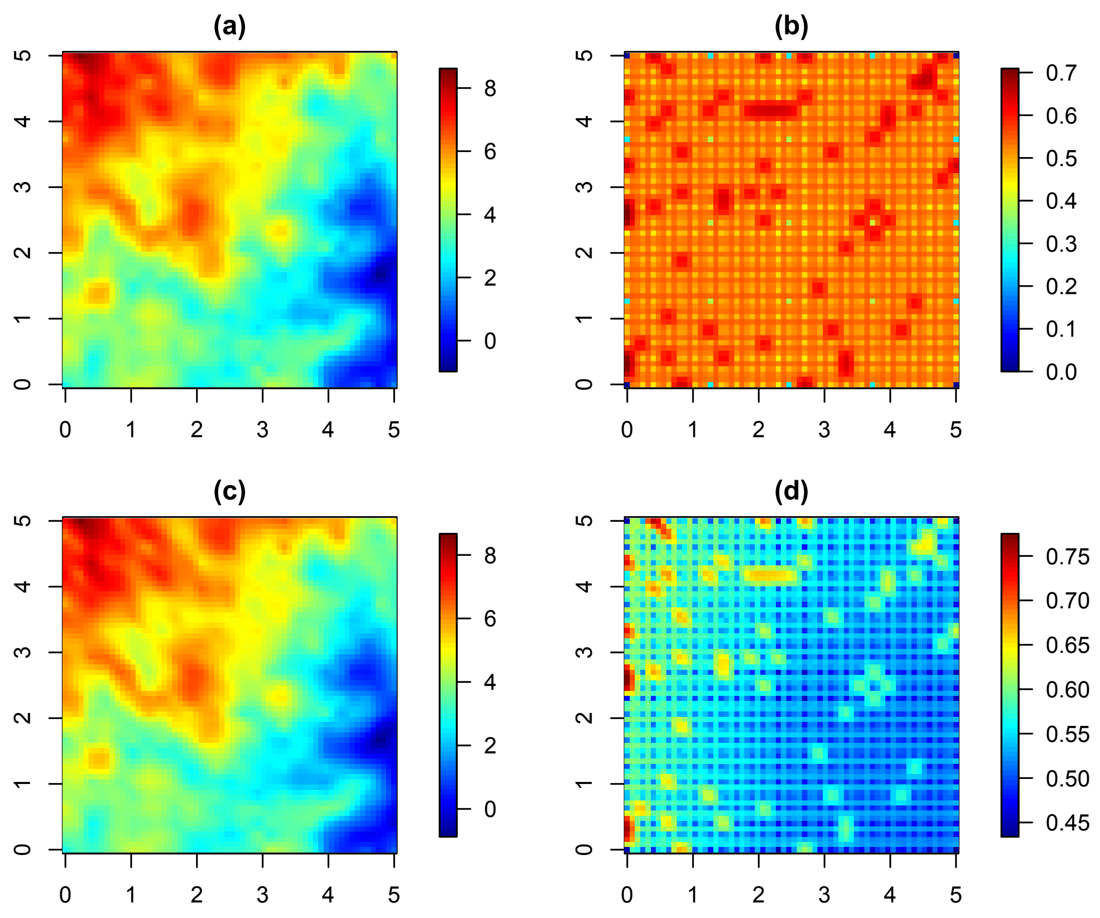

Figure 2: Predictions and prediction errors from the stationary model (a. and b.) and the nonstationary model (c. and d.).

```
> plot.preds( locations = pred.locs, predmeans = pred.aniso$pred.means,
+             predSDs = pred.aniso$pred.SDs )
> plot.preds( locations = pred.locs, predmeans = pred.NS$pred.means,
+             predSDs = pred.NS$pred.SDs )
```

and these plots are provided in Figure 2.

To visualize the locally-estimated anisotropy, the mixture component kernel matrices can be plotted along with the stationary anisotropy ellipse using

```
> plot.mc( NSfit.model$mc.locations, NSfit.model$mc.kernels,
+             fit.radius = 2.3, aniso.mat = anisofit.model$aniso.mat )
```

However, for the simulated data, since we know what the true ellipses are, we can overlay the truth on top of what is estimated. This is given in Figure 3, which plots the mixture component locations

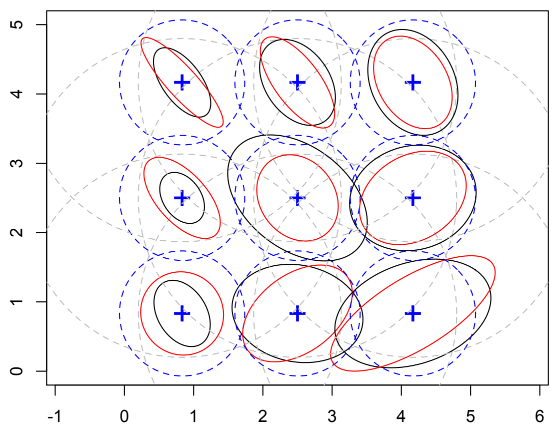

Figure 3: True mixture component ellipses (red) with fit radius (gray), nonstationary ellipses (black), and the stationary ellipse (blue).

along with the true anisotropy ellipse (red), estimation region (dashed circle), estimated anisotropy ellipse (solid black), and the stationary ellipse (blue).

As an additional visualization of the estimated nonstationarity, estimated correlation plots for a particular reference point can be obtained by

```
> par(mfrow=c(1,2))
> plot.correlation( model.fit.obj = NSfit.model,
+                   ref.loc = c(1.5, 3.5),
+                   all.pred.locs = pred.locs,
+                   NS.model = TRUE, grid = TRUE )
> plot.correlation( model.fit.obj = anisofit.model,
+                   ref.loc = c(1.5, 3.5),
+                   all.pred.locs = pred.locs,
+                   NS.model = FALSE, grid = TRUE )
```

and are given in Figure 4, for both the nonstationary and stationary models, highlighting the non-elliptical nature of the estimated correlation patterns under the nonstationary model. For comparison, the true correlation has also been plotted in Figure 4.

Note that the nonstationary model outperforms the stationary model in terms of both CRPS and MSPE.

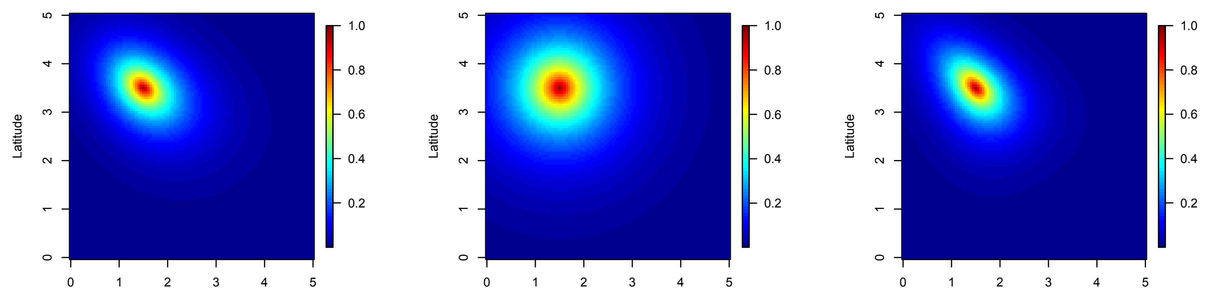

Figure 4: Estimated correlations for a reference point, showing the nonstationary (left) and stationary (center) models, as well as the true correlation (right).

7 Real data example: Annual precipitation

The nonstationary model is also applied to a moderately large, real data set, consisting of the total annual precipitation in the western United States for 1997. The data is available online from the National Center for Atmospheric Research (`http://www.image.ucar.edu/GSP/Data/US.monthly.met`) as part of a larger data set that includes measurements for the entire United States. For the purposes of this analysis, a subset of the data that includes the western United States was chosen (see Figure 5) because precipitation is more smooth and densely observed over the central and eastern United States. While the data appears to be gridded, it is in fact point-level data – the gridded appearance is due to the plotting. This subset is included in the package as an `RData` file and consists of 1446 observations.

A total of six spatial models were fit to this dataset, the details of which are summarized in Table 2: two stationary models and four nonstationary models. All six models had the same mean structure, which included the main effects of longitude and latitude as well as an intercept. Fifteen percent of the observations ($m = 216$) were held out as a test data set in order to evaluate each model, leaving $n = 1230$ observations as a training data set. The two stationary models were fit using the `Aniso.fit` function, using both options of `iso.model`, and the nonstationary models were fit using the `NSconvo.fit` function. For each of the nonstationary models, the model was fit multiple times using different sets of mixture component locations and fit radii. The specific

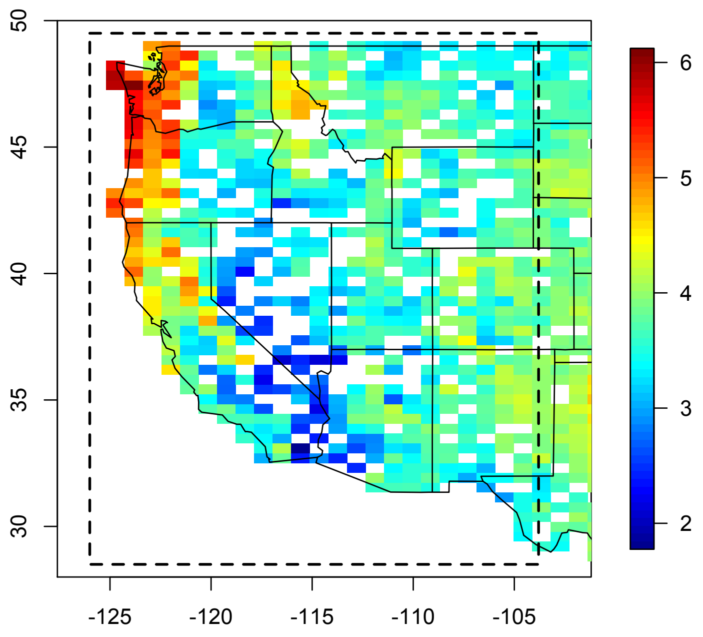

Figure 5: Annual precipitation for 1997 (log mm), with the estimation region inside the dashed line. Note that the data is in fact point-level data, in spite of its gridded appearance.

models summarized in Table 2 were chosen based on the evaluation criteria; the number of mixture component locations and fit radii are summarized in Table 3.

Parameter estimates for each of the models are summarized in Table 3. Note that the mean parameters are very sensitive to which of the variance/covariance parameters are allowed to be spatially-varying, as is the smoothness parameter.

Table 2 also provides the values of the evaluation criteria for each model. While the gains are small, the nonstationary models all outperform the stationary models in terms of both MSPE and CRPS; of the nonstationary models, NS3 provides the best performance. As a result, the nonstationary model NS3 was chosen as the preferred model for the following visual summaries; the stationary model S1 is used for comparison.

As an initial summary of the nonstationary model, consider the locally-estimated anisotropy ellipses for each of the mixture component locations, shown in Figure 6. Also shown is the ellipse corresponding to the isotropic model. Next, consider the predictions and prediction standard errors for the stationary model S1 and the nonstationary model NS3, shown in Figure 7. Figure

Label	**Covariance model details**	**CRPS**	**MSPE**	**Comp. time (min)**
S1	Isotropic, constant nugget and variance	0.1483	0.0758	3.76
S2	Anisotropic, constant nugget and variance	0.1546	0.0752	10.60
NS1	SV anisotropy, constant nugget and variance	0.1303	0.0719	69.26
NS2	SV anisotropy and variance, constant nugget	0.1365	0.0704	60.26
NS3	SV anisotropy and nugget, constant variance	0.1232	0.0719	49.54
NS4	SV anisotropy, variance, and nugget	0.1275	0.0711	40.95

Table 2: A brief summary of the different models fit to the precipitation data, along with evaluation criteria for the holdout set and computational time. Note: "SV" indicates "spatially-varying." [Add computational details.]

Parameter	**S1**	**S2**	**NS1**	**NS2**	**NS3**	**NS4**
β_0 (int.)	-3.080	-2.857	-0.333	1.451	-0.168	1.503
β_1 (long.)	-0.054	-0.051	-0.031	-0.020	-0.029	-0.020
β_2 (lat.)	0.021	0.023	0.019	0.002	0.019	0.002
λ_1	2.95	2.56	SV	SV	SV	SV
λ_2	2.95	3.49	SV	SV	SV	SV
η	0.00	1.359	SV	SV	SV	SV
τ^2	0.0129	0.0127	0.0276	0.0196	SV	SV
σ^2	0.491	0.431	0.2238	SV	0.206	SV
ν	0.5	0.5	0.617	0.589	0.405	0.466
K	–	–	15	15	15	15
λ_w	–	–	3.5	3.5	3.5	3.5
r	–	–	4.95	4.95	4.95	4.95

Table 3: Parameter estimates for the five spatial models fit to the precipitation data set, indicating which parameters are spatially-varying (SV). Also given are the number of mixture component locations (K), the tuning parameter for the weight function (λ_w), and the fitting radius (r).

8 contains estimates of the spatially-varying nugget variance (τ^2) and process variance (σ^2) over the spatial region of interest. Finally, correlation plots for three reference points are given in Figure 9, comparing the stationary and nonstationary model. Again, we are able to notice that the nonstationary model is able to capture different spatial dependence patterns over space, unlike the stationary model.

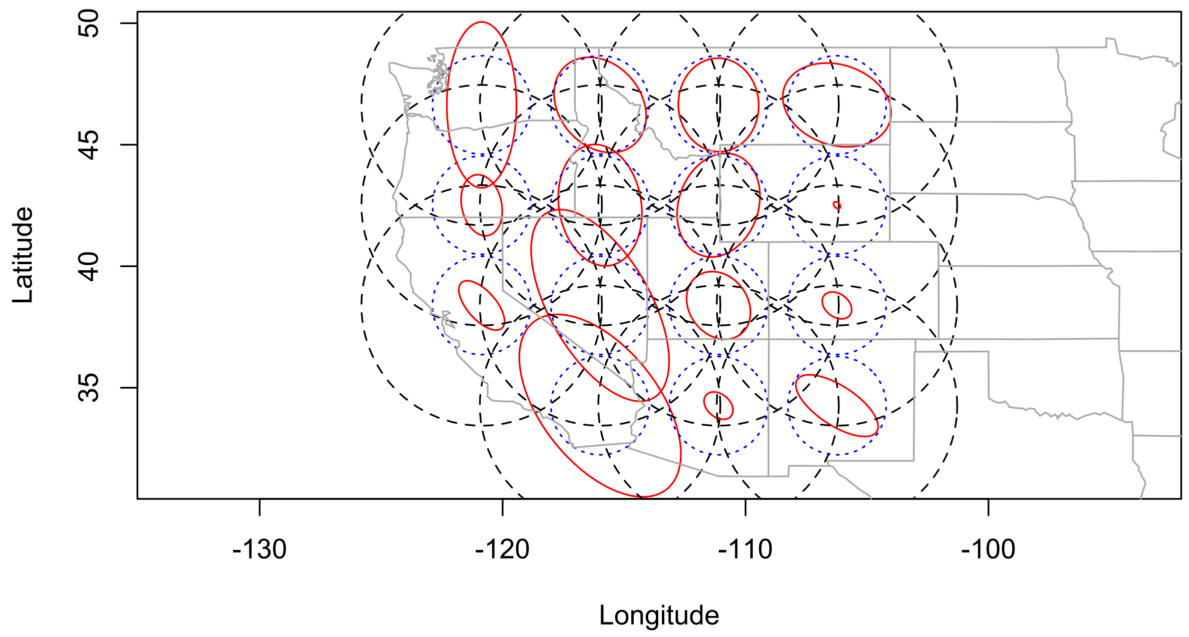

Figure 6: Estimated mixture component ellipses for the nonstationary model (red), the isotropic model (blue), and the estimation region (dashed black).

8 Discussion

In this paper, we have presented a new nonstationary spatial Gaussian process model that is highly flexible yet computationally feasible, as shown through its implementation in the new `convoSPAT` package for `R`. The model allows for visualization of the estimated nonstationarity, for both the spatially-varying parameters and the estimated spatial correlation.

The tradeoff for the computational tractability of this model is that the uncertainty in the locally-estimated parameters is not accounted for in global estimation.

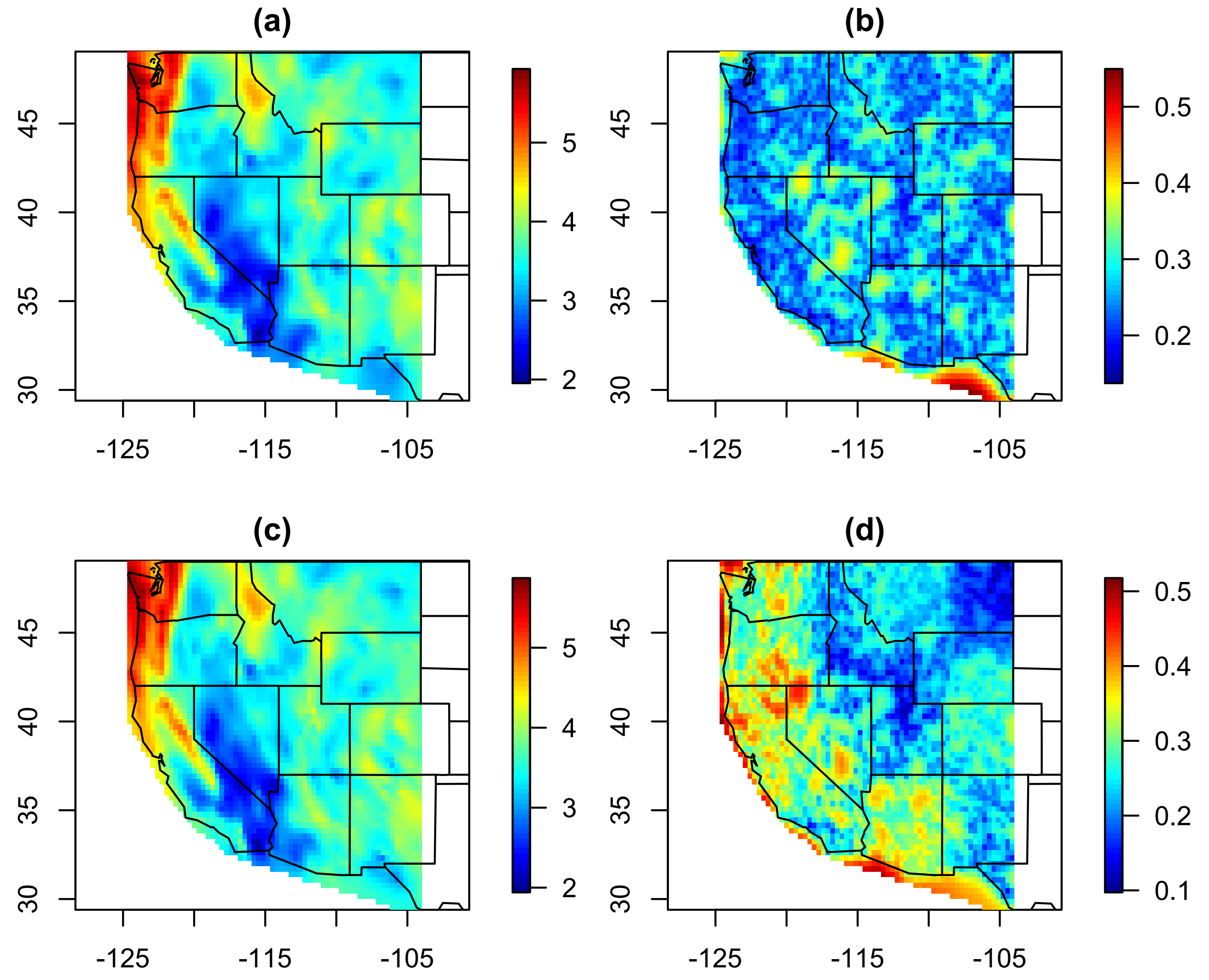

Figure 7: Predictions and prediction standard errors for the stationary model S1 (plots (a) and (b)) and the nonstationary model NS4 (plots (c) and (d)).

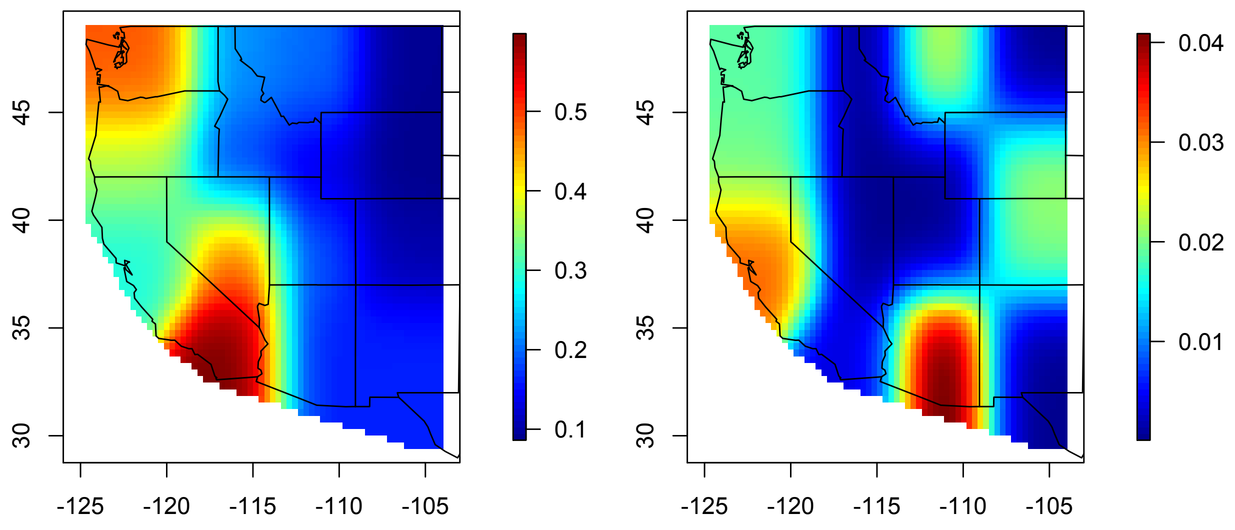
Figure 8: Plots of the estimated spatially-varying process variance σ^2 (left) and nugget variance τ^2 (right) for model NS4.

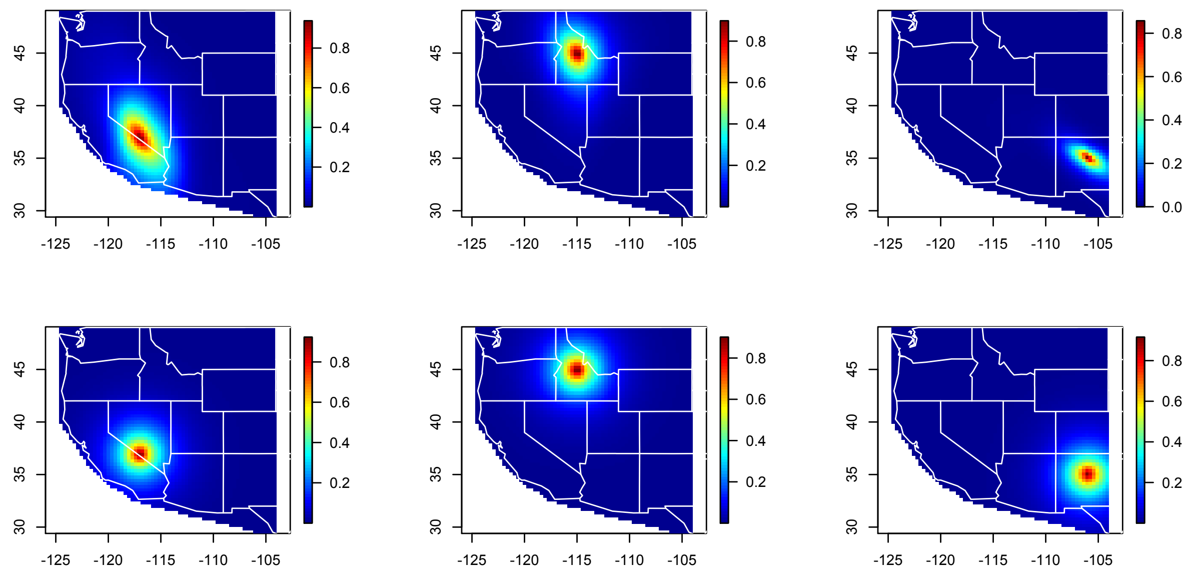
Figure 9: Correlation plots for three reference points, comparing the nonstationary model NS4 (top) and stationary model S1 (bottom).

References

Anderes, E. B. and Stein, M. L. (2011). Local likelihood estimation for nonstationary random fields. *Journal of Multivariate Analysis*, 102(3):506 – 520.

Barry, R. P. and Ver Hoef, J. M. (1996). Blackbox kriging: Spatial prediction without specifying variogram models. *Journal of Agricultural, Biological, and Environmental Statistics*, 1(3):297–322.

Calder, C. A. (2008). A dynamic process convolution approach to modeling ambient particulate matter concentrations. *Environmetrics*, 19(1):39–48.

Damian, D., Sampson, P. D., and Guttorp, P. (2001). Bayesian estimation of semi-parametric non-stationary spatial covariance structures. *Environmetrics*, 12(2):161–178.

Fuentes, M. (2001). A high frequency kriging approach for non-stationary environmental processes. *Environmetrics*, 12(5):469–483.

Fuentes, M. (2002). Spectral methods for nonstationary spatial processes. *Biometrika*, 89(1):197–210.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Higdon, D. (1998). A process-convolution approach to modelling temperatures in the North Atlantic Ocean. *Environmental and Ecological Statistics*, 5(2):173–190.

Higdon, D. (2002). Space and space-time modeling using process convolutions. In Anderson, C., Barnett, V., Chatwin, P., and El-Shaarawi, A., editors, *Quantitative Methods for Current Environmental Issues*, pages 37–56. Springer London.

Hoef, J. M. V. and Barry, R. P. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275 – 294.

Ickstadt, K. and Wolpert, R. L. (1998). Spatial regression for marked point processes. *Bayesian Statistics*, 6.

Katzfuss, M. (2013). Bayesian nonstationary spatial modeling for very large datasets. *Environmetrics*, 24(3):189–200.

Kitanidis, P. K. (1983). Statistical estimation of polynomial generalized covariance functions and hydrologic applications. *Water Resources Research*, 19(4):909–921.

Kleiber, W. and Nychka, D. (2012). Nonstationary modeling for multivariate spatial processes. *Journal of Multivariate Analysis*, 112(0):76 – 91.

Nychka, D., Furrer, R., and Sain, S. (2014). *fields: Tools for spatial data*. R package version 7.1.

Paciorek, C. J. (2003). *Nonstationary Gaussian processes for regression and spatial modelling*. PhD thesis, Carnegie Mellon University.

Paciorek, C. J. and Schervish, M. J. (2006). Spatial modeling using a new class of nonstationary covariance functions. *Environmetrics*, 17:483–506.

Patterson, H. D. and Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3):545–554.

Patterson, H. D. and Thompson, R. (1974). Maximum likelihood estimation of components of variance. Proc. Eighth International Biochem. Conf.

Reich, B. J., Eidsvik, J., Guindani, M., Nail, A. J., and Schmidt, A. M. (2011). A class of covariate-dependent spatiotemporal covariance functions for the analysis of daily ozone concentration. *The Annals of Applied Statistics*, 5(4):2425–2447.

Ribeiro Jr., P. J. and Diggle, P. J. (2001). geoR: a package for geostatistical analysis. *R-NEWS*, 1(2):14–18. ISSN 1609-3631.

Risser, M. and Calder, C. (2014). Regression-based covariance functions for nonstationary spatial modeling. Accessed at http://arxiv.org/abs/1410.1494.

Sampson, P. D. and Guttorp, P. (1992). Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119.

Schmidt, A. M., Guttorp, P., and O'Hagan, A. (2011). Considering covariates in the covariance structure of spatial processes. *Environmetrics*, 22(4):487–500.

Schmidt, A. M. and O'Hagan, A. (2003). Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758.

Stein, M. L. (2005). Nonstationary spatial covariance functions. *Unpublished technical report*.

Thiébaux, H. J. (1976). Anisotropic correlation functions for objective analysis. *Monthly Weather Review*, 104:994–1002.

Thiébaux, H. J. and Pedder, M. A. (1987). *Spatial Objective Analysis: with applications in atmospheric science*. Academic Press.

Tibshirani, R. and Hastie, T. (1987). Local likelihood estimation. *Journal of the American Statistical Association*, 82(398):pp. 559–567.

Ver Hoef, J. M., Cressie, N., and Barry, R. P. (2004). Flexible spatial models for kriging and cokriging using moving averages and the fast fourier transform (fft). *Journal of Computational and Graphical Statistics*, 13(2):265–282.

Vianna Neto, J. H., Schmidt, A. M., and Guttorp, P. (2014). Accounting for spatially varying directional effects in spatial covariance structures. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 63(1):103–122.