SSNbayes: An R Package for Bayesian Spatio-Temporal Modelling on Stream Networks

by Edgar Santos-Fernandez, Jay M. Ver Hoef, James McGree, Daniel J. Isaak, Kerrie Mengersen, and Erin E. Peterson

Abstract Spatio-temporal models are widely used in many research areas from ecology to epidemiology. However, a limited number of computational tools are available for modeling river network datasets in space and time. In this paper, we introduce the R package SSNbayes for fitting Bayesian spatio-temporal models and making predictions on branching stream networks. SSNbayes provides a linear regression framework with multiple options for incorporating spatial and temporal autocorrelation. Spatial dependence is captured using stream distance and flow connectivity while temporal autocorrelation is modelled using vector autoregression approaches. SSNbayes provides the functionality to make predictions across the whole network, compute exceedance probabilities, and other probabilistic estimates, such as the proportion of suitable habitat. We illustrate the functionality of the package using a stream temperature dataset collected in the Clearwater River Basin, USA.

1 Introduction

Rivers and streams are of vital ecological and economic importance (Vörösmarty et al., 2010) but are under pressure from anthropogenic impacts such as climate change, pollution, water extractions, and overfishing. In the past, data describing critical characteristics such as nutrients, sediments, pollutants and stream flow tended to be sparse in space and/or time. However, recent developments in in-situ sensor technology have revolutionized ecological research and natural resource monitoring. These new data sets create exciting opportunities to measure, learn about, and manage the spatio-temporal dynamics of abiotic (e.g. temperature, water chemistry, habitat characteristics) and biotic processes (e.g. migration, predation, and competition). However, the unique spatial relationships found in stream data (e.g. branching network structure, longitudinal (upstream/downstream) connectivity, water flow volume and direction) and high-frequency of sampling create analytical challenges that make it difficult to gain meaningful insights from these datasets. We attempt to overcome these challenges through the **SSNbayes** package which provides convenient and practical tools to undertake Bayesian inference in complex spatial and temporal stream network settings.

1.1 Motivating dataset: repeated measures from *in-situ* sensor locations in a river

Consider the river network in the Clearwater River Basin, USA shown in Fig 1. Water temperature data were collected at fixed time intervals using in-situ sensors placed at 18 unique locations throughout the network (Isaak et al., 2018). We want to use these data to address several research questions and goals. Firstly, we would like to analyse water temperature to assess the impact of covariates such as air temperature. Secondly, we aim to make predictions with uncertainty at other locations throughout the network (approximately every 1 km). Also, we want to impute missing data e.g. the most downstream point in Fig 1 (location 12), and determine regions of the network which remain suitable habitats over time to sustain fish species such as bull trout which typically preferred water temperatures of < 13 °C. Throughout this paper, we show how the **SSNbayes** package can be used to analyse these spatial and temporal data, and address these motivating research questions.

1.2 A brief review

A number of R software packages for spatial stream-network modelling have been developed over the last few decades (Ver Hoef et al., 2014; Skoien et al., 2014; Rushworth, 2017). These packages account for unique spatial relationship found in stream data. For example, the R packages **SSN** and **SSN2** (Ver Hoef et al., 2014; Dumelle et al., 2023) fits spatial regression models for stream networks, with autocorrelation in time only possible by using random effects as repeated measures, which induces equal correlation for all times at a location. Similarly, spatial additive models can be fitted using the package **smnet** (Rushworth, 2017). However, these models are not designed to simultaneously account for the temporal variability that often accompanies spatial variation in new data sets derived from modern sensor arrays.

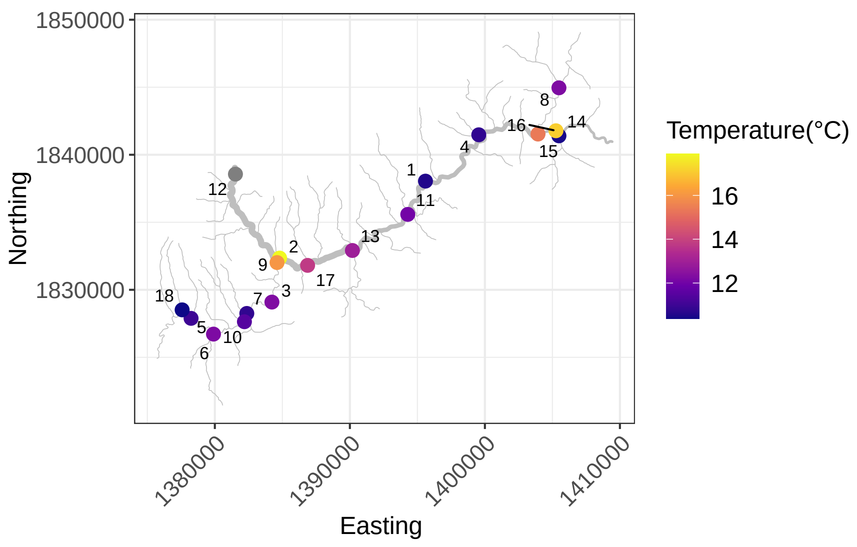

Figure 1: Mean daily water temperature in degrees Celsius on August 1, 2012, at 18 different spatial locations in the Clearwater stream network in the US. The temperature values are represented by a color scale, with cooler temperatures shown in blue and warmer temperatures shown in yellow. Each spatial location is labeled with a unique identifier (`locID`). The plot highlights the variation in water temperature across the different locations in the network.

There are several R packages for spatio-temporal modelling that are described in the Space-time CRAN Task View (Pebesma, 2021). For example, spatial/temporal dependence can be incorporated via the **nlme** package package (Pinheiro and Bates, 2000; Pinheiro et al., 2020). Other packages such as **spBayes** (Finley et al., 2015) allow random effects modelling for point-referenced data. **CARBayes** (Lee, 2013) contains useful tools for implementing Bayesian spatial models using random effects via conditional autoregressive (CAR) priors. Similarly, spatial process data can be represented using kernels e.g. using the package **RandomFields** (Schlather et al., 2015). Interpolation of spatial data and Kriging can be done via tools from the package **geoR** (Ribeiro Jr et al., 2020). One of the most popular implementations among practitioners is the **R-INLA** package (Lindgren and Rue, 2015), which uses approximate Bayesian inference via integrated nested Laplace approximations. Multiple latent Gaussian spatio-temporal models can be fit in **R-INLA**. **FRK** (Zammit-Mangion and Cressie, 2021) makes use of spatial basis functions and discrete areal units with a focus on large datasets. In the same line, new packages harnessing the power of **rstan** (Carpenter et al., 2017) are also emerging. For instance, **bmstdr** (Sahu et al., 2022) implements several spatio-temporal approaches for point referenced and areal unit datasets, and **geostan** (Donegan, 2022) fits spatial models for areal data. However, none of these packages are specifically designed to account for the unique spatial relationships found in data collected on streams.

Here, we describe the **SSNbayes** package which has been designed to address many of the limitations of the current software tools for spatio-temporal modelling on stream networks. More specifically, **SSNbayes** can fit spatio-temporal stream-network models and produce predictions in space and time, with associated estimates of uncertainty. It uses the Bayesian inference machinery implemented using the probabilistic programming language Stan.

The rest of the paper is organised as follows: we introduce the relevant statistical models in the Methods section, followed by an overview of the package structure and functions. We then demonstrate how the **SSNbayes** package can be used to explore, analyse and draw conclusions from a stream temperature data set collected in the western United States (USA). A second reproducible example is provided in the Appendix to help users adapt the R code for their own data. Finally, we conclude with a discussion of the benefits and challenges in using the **SSNbayes** package and Bayesian spatio-temporal models on stream networks, as well as potential extensions to the methods and improvements to the package.

2 Methods

Data collected on stream networks often exhibit complex patterns of spatial autocorrelation resulting from ecosystem processes occurring within branching stream networks, as well as between the aquatic and terrestrial environments (Peterson et al., 2013). It is therefore common for streams data to exhibit *both* Euclidean and in-stream patterns of spatial autocorrelation at multiple scales (Peterson et al., 2006; Peterson and Ver Hoef, 2010). Thus, we start this section with a description of spatial models based on Euclidean distance and then describe methodological extensions to stream networks. These models provide the foundation to describe the Bayesian hierarchical spatio-temporal model variations implemented in the **SSNbayes** package.

General space-time model

Consider the general spatio-temporal linear model:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{v} + \boldsymbol{\epsilon}, \tag{1}$$

where the response variable $\boldsymbol{y} = [\boldsymbol{y}_1^{'}, \boldsymbol{y}_2^{'}, \cdots, \boldsymbol{y}_T^{'}]^{'}$ is a stacked vector of length $n = S \times T$ for S spatial locations and T time points. We order data such that vector $\boldsymbol{y}_t$ contains the observations at the S spatial locations at time t for $t = 1, \ldots, T$. Let $\boldsymbol{X_t}$ be an $S \times p$ design matrix of p covariates for the tth time. We construct a stacked matrix of covariates $\boldsymbol{X} = [\boldsymbol{X}_1^{'}, \boldsymbol{X}_2^{'}, \cdots, \boldsymbol{X}_T^{'}]^{'}$ with dimensions $n \times p$. We then define $\boldsymbol{\beta}$ as a $p \times 1$ vector of regression coefficients. We let $\boldsymbol{v} = [\boldsymbol{v}_1^{'}, \boldsymbol{v}_2^{'}, \cdots, \boldsymbol{v}_T^{'}]^{'}$ be a stacked vector of n spatial random effects, where $\boldsymbol{v}_t$ is a vector of length S for each t, and all $\boldsymbol{v}_t$ have the same spatial dependence model (shared locations and parameters across all times t). For example, $\boldsymbol{v}_t$ can be modelled using a Gaussian process (Banerjee et al., 2014), but we also use spatial stream-network models that we describe below for each $\boldsymbol{v}_t$. The final step in our construction is to use vector autoregressive models to add the temporal components which we develop below. The vector $\boldsymbol{\epsilon}$ is the independent unstructured random error term, where $\text{var}(\boldsymbol{\epsilon}) = \sigma_0^2 \boldsymbol{I}$. The parameter σ_0^2 is called the nugget effect and $\boldsymbol{I}$ is an $n \times n$ identity matrix.

Euclidean distance models

A typical modelling approach to capture spatial dependence is via the second moment of $\boldsymbol{v}$ from Eq 1 where the amount of autocorrelation decays with the Euclidean distance. Some of the most common covariance functions are the exponential, Gaussian and spherical (Cressie and Wikle, 2015; Banerjee et al., 2014):

$$\text{exponential model,}\;\; C_{ED}(d \mid \boldsymbol{\theta}) = \sigma_e^2 e^{-3d/\alpha_e}, \tag{2}$$

$$\text{Gaussian model,}\;\; C_{ED}(d \mid \boldsymbol{\theta}) = \sigma_e^2 e^{-3(d/\alpha_e)^2}, \tag{3}$$

and

$$\text{spherical model,}\;\; C_{ED}(d \mid \boldsymbol{\theta}) = \sigma_e^2 \left(1 - \frac{3d}{2\alpha_e} + \frac{d^3}{2\alpha_e^3}\right) \mathbb{1}(d/\alpha_e \leqslant 1), \tag{4}$$

where $\alpha_e \in (0, \infty), \sigma_e^2 > 0$, and d is the Euclidean distance between two locations s_i and s_j. The vector $\boldsymbol{\theta}$ represents the spatial parameters (α_e, σ_e^2), where σ_e^2 is the partial sill, α_e is the spatial range parameter and $\mathbb{1}(\cdot)$ is the indicator function. The partial sill is the resulting variance after accounting for the nugget effect (sill minus nugget effect). Negligible spatial correlation is assumed between points located at a distance greater than the spatial range parameter.

2.1 Spatial models for stream networks

The stream network data shown in Fig.2 represents repeated measures at several time points t from four spatial locations (s_1 to s_4). Stream distance is defined as the separation distance between two locations when movement is restricted to the network. The direction of the water flow is also shown in the figure (from north to south) and the stream outlet (i.e. most downstream point on the stream network) is below location s_4. The watershed (i.e. drainage basin) includes the land that contributes water flow to a discrete downstream location in the stream network. Thus, the watershed for the stream outlet in Fig. 2 includes all of the upstream regions (r_1-r_4). Spatial locations s_1 and s_3 are considered flow-connected because the water flows from the upstream location s_1 to the downstream location s_3. In contrast, s_1 and s_2 are considered flow-unconnected because they reside on the same stream network, but do not share flow.

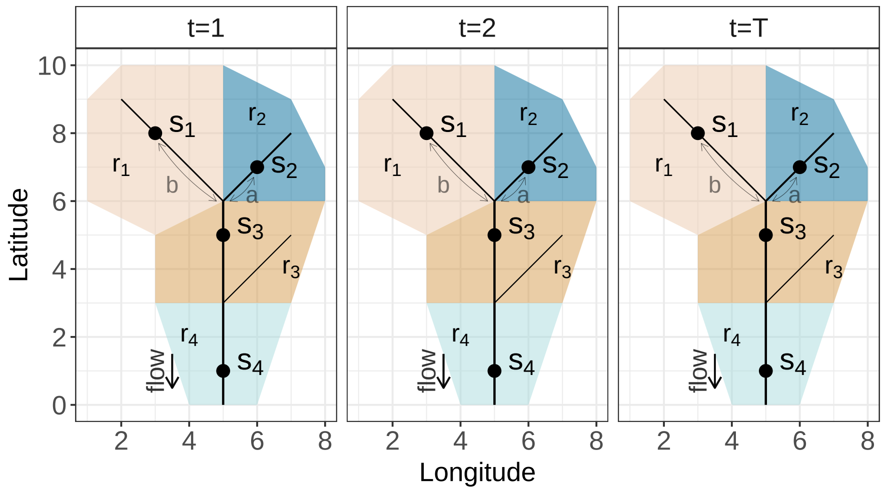

Figure 2: Visualization of a stream network at multiple time points, with water flowing from top to bottom. The plot displays four spatial locations ($s_1 - s_4$) and four regions ($r_1 - r_4$), with the colors indicating the regions. This visualization allows for an understanding of the spatial and temporal dynamics of the stream network.

Multiple covariance models have been proposed for stream networks that describe the unique spatial dependence related to network structure and stream flow. These models are based on moving average constructions that use stream distance, and were introduced by Ver Hoef et al. (2006) and Cressie et al. (2006) (tail-up models) and Ver Hoef and Peterson (2010) and Garreta et al. (2010) (tail-down models); several models of each class are given below. These spatial covariance matrices have been extensively employed in numerous water quality modelling frameworks and applications (Money et al., 2009a,b; Isaak et al., 2014; McManus et al., 2020; Jackson et al., 2018; Rodríguez-González et al., 2019) and are described in more detail below.

Tail-up models

Tail-up covariance models were developed by convolving a moving average function with white noise (Ver Hoef et al., 2006). As the name suggests, the moving average function points in the upstream direction from a stream location in a tail-up model, which restricts correlation to flow-connected locations only. In addition, the function must be split at upstream junctions using spatial weights, which maintain stationary variances by controlling the proportion allocated to each upstream segment.

Given a pair of sites s_i and s_j, the tail-up covariance matrix is defined as:

$$C_{TU}(s_i, s_j|\boldsymbol{\theta}) = \begin{cases} 0 & \text{if } s_i, s_j \text{ are flow-unconnected,} \\ C_u(h \mid \boldsymbol{\theta})W_{ij} & \text{if } s_i, s_j \text{ are flow-connected,} \end{cases}$$

where $C_u(h \mid \boldsymbol{\theta})$ is an unweighted tail-up covariance between two locations based on the total stream distance between them, h. W_{ij} represents the spatial weights between sites i and j. C_u can take a variety of forms including:

$$\text{Tail-up exponential model,} \quad C_u(h \mid \boldsymbol{\theta}) = \sigma_u^2 e^{-3h/\alpha_u}, \tag{5}$$

$$\text{Tail-up linear-with-sill model,} \quad C_u(h \mid \boldsymbol{\theta}) = \sigma_u^2(1 - h/\alpha_u)\mathbb{1}(h/\alpha_u \leqslant 1), \tag{6}$$

$$\text{Tail-up spherical model,} \quad C_u(h \mid \boldsymbol{\theta}) = \sigma_u^2 \left(1 - \frac{3h}{2\alpha_u} + \frac{h^3}{2\alpha_u^3}\right) \mathbb{1}(h/\alpha_u \leqslant 1) \tag{7}$$

where σ_u^2 is the partial sill and α_u is the range parameter.

The spatial weights used in the tail-up model are generated for flow-connected locations

$$W_{ij} = \sqrt{\frac{\Omega(s_j)}{\Omega(s_i)}},$$

where $\Omega(s_i)$ and $\Omega(s_j)$ are the additive function values for the upstream and downstream site, respectively. Additive function values can be derived from any variable available for every line segment in a stream network, but are often based on an ecologically meaningful variable (e.g. stream flow) or a surrogate (e.g. watershed area), which is thought to represent relative influence in a stream network (Frieden et al., 2014). Additive function values can be generated in **SSN** using the `additive.function()` or in the STARS ArcGIS custom toolset using the `SegmentPI`, `Additive Function - Edges`, and `Additive Function - Sites` tools (Peterson and Ver Hoef, 2014). For additional information about how the additive function values are constructed, please see Ver Hoef et al. (2006) or Appendices A in Santos-Fernandez et al. (2022) or Peterson and Ver Hoef (2010).

Tail-down models

Tail-down models are also based on a moving average contruction (Ver Hoef and Peterson, 2010), but in this case the function points in the downstream direction. This allows the models to describe spatial correlation between both flow-connected and flow-unconnected locations. In addition, the moving average functions converge downstream and do not need to be split at junctions using spatial weights.

There is a distinction between flow-connected and flow-unconnected relationships in a tail-down model. When pairs of sites are flow-connected, distance is based on the total stream distance between them, h. When sites are flow-unconnected (e.g. s_1 and s_2 in Fig.2), we define a and b as the stream distance from s_1 and s_2 to their first common downstream junction, so that $a \leqslant b$.

Tail-down exponential model,

$$C_{TD}(a,b,h|\boldsymbol{\theta}) = \begin{cases} \sigma_d^2 e^{-3h/\alpha_d} & \text{if flow-connected,} \\ \sigma_d^2 e^{-3(a+b)/\alpha_d} & \text{if flow-unconnected,} \end{cases}$$

Tail-down linear-with-sill model,

$$C_{TD}(a,b,h|\theta) = \begin{cases} \sigma_d^2(1-\frac{h}{\alpha_d})\mathbb{1}(\frac{h}{\alpha_d} \leqslant 1) & \text{if flow-connected,} \\ \sigma_d^2(1-\frac{b}{\alpha_d})\mathbb{1}(\frac{b}{\alpha_d} \leqslant 1) & \text{if flow-unconnected,} \end{cases}$$

Tail-down spherical model,

$$C_{TD}(a,b,h|\boldsymbol{\theta}) = \begin{cases} \sigma_d^2(1-\frac{3h}{2\alpha_d}+\frac{h^3}{2\alpha_d^3})\mathbb{1}(\frac{h}{\alpha_d} \leqslant 1) & \text{if flow-connected,} \\ \sigma_d^2(1-\frac{3a}{2\alpha_d}+\frac{b}{2\alpha_d})(1-\frac{b}{\alpha_d})\mathbb{1}(\frac{b}{\alpha_d} \leqslant 1) & \text{if flow-unconnected,} \end{cases}$$

where σ_d^2 is the partial sill, α_d is the range parameter, and $\mathbb{1}(\cdot)$ is the indicator function, equal to 1 if its argument is true, otherwise it is zero.

2.2 Mixed models

A mixture of Euclidean, tail-up and tail-down covariance matrices is often used to capture the complex patterns of spatial dependency found in stream data. In Eq 1, $\boldsymbol{v}_t$ for a purely spatial case is a vector of dimension S corresponding to the spatial locations, with covariance matrix $\boldsymbol{\Sigma}$.

$$\boldsymbol{\Sigma} = COV(\boldsymbol{v}) = \boldsymbol{C}_{ED} + \boldsymbol{C}_{TU} + \boldsymbol{C}_{TD} = \sigma_e^2\boldsymbol{R}_e(\alpha_e) + \sigma_u^2\boldsymbol{R}_u(\alpha_u) + \sigma_d^2\boldsymbol{R}_d(\alpha_d), \tag{8}$$

where σ_e^2, σ_u^2, and σ_d^2 are the partial sills for Euclidean, tail-up and tail-down functions, respectively. The correlation matrices $\boldsymbol{R}_u(\alpha_u)$, $\boldsymbol{R}_d(\alpha_d)$ and $\boldsymbol{R}_e(\alpha_e)$ are obtained as a function of the range parameters α_u, α_d and α_e (Ver Hoef et al., 2014).

For space-time applications, we can use the same spatial covariance matrix or we can build a dynamic model with spatial parameters that are time-specific. We opted for the first approach in the **SSNbayes**, since this reduces the number of parameters to be estimated from the model and is less computationally demanding. We return to this point in the Discussion.

2.3 Spatio-temporal stream network models

Following the above discussion, consider the stream network in Fig.2, that evolves across discrete time points $t = 1, 2, \ldots, T$. Let a response variable $\boldsymbol{y}_t$ be an $S \times 1$ vector of random variables at unique and fixed spatial locations of $s = 1, 2, \ldots, S$. We start with the following conditional spatio-temporal model:

$$[\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T] = \prod_{t=2}^{T} [\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{\theta}, \mathbf{X}_t, \mathbf{X}_{t-1}, \boldsymbol{\beta}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}] \tag{9}$$

where $\boldsymbol{y}_1$ is the process at $t = 1$, and

$$[\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{\theta}, \mathbf{X}_t, \mathbf{X}_{t-1}, \boldsymbol{\beta}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}] \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma} + \sigma_0^2 \boldsymbol{I}), \tag{10}$$

Here, $\boldsymbol{\Sigma} = COV(\boldsymbol{v}_t)$ is an $S \times S$ spatial covariance matrix of the form given in Eq 8. The mean $\boldsymbol{\mu}_t$ can be expressed as a vector autoregressive process of order one VAR(1) (Hamilton, 1994):

$$\boldsymbol{\mu}_t = \mathbf{X}_t \boldsymbol{\beta} + \boldsymbol{\Phi}(\boldsymbol{y}_{t-1} - \mathbf{X}_{t-1}\boldsymbol{\beta}), \tag{11}$$

The square transition matrix, $\boldsymbol{\Phi}$ of dimension $S \times S$, has elements ϕ_{ij}, which describe the conditional temporal cross-correlation between two spatial locations i and j.

2.4 Vector autoregressive model variations

We use a vector autoregressive (VAR) approach to simultaneously model time series from multiple spatial locations in the network. This stochastic approach allows capturing and incorporating temporal dependence across multiple time series. Two variations of the vector autoregressive spatial process have been implemented in the **SSNbayes** package.

Case 1 (AR)

The simplest case considers the same temporal autocorrelation for all spatial locations. Therefore all the diagonal elements of $\boldsymbol{\Phi}$ are equal to ϕ and all the off-diagonal elements are set to zero, which assumes negligible cross-correlation between time series. That is:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi & 0 & \cdots & 0 \\ 0 & \phi & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \phi \end{bmatrix}. \tag{12}$$

Under this model, the temporal autocorrelation for a fixed spatial location is an AR1 model with autocorrelation parameter ϕ, and the joint spatio-temporal autocovariance matrix is the separable model,

$$\text{var}(\mathbf{y}) = \boldsymbol{C}_{OO} = \frac{1}{1-\phi^2} \begin{bmatrix} 1 & \phi & \cdots & \phi^{T-1} \\ \phi & 1 & \cdots & \phi^{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ \phi^{T-1} & \phi^{T-2} & \cdots & 1 \end{bmatrix} \otimes (\boldsymbol{\Sigma} + \sigma_0^2 \boldsymbol{I}), \tag{13}$$

where $\otimes$ is the Kronecker product.

Spatial locations in large river networks often have different elevations, climatic conditions, or local flow regimes and this can affect the amount of temporal autocorrelation in observations. Hence, the assumption that there is a common ϕ for all locations may not always be appropriate and this motivates Case 2.

Case 2 (VAR)

In the second method, rather than a shared ϕ across all locations, each location gets its own temporal autocorrelation parameter ϕ_s for $s \in 1, \cdots, S$. This is known as the autoregressive shock model (Wikle et al., 1998), which can be defined through $\boldsymbol{\Phi}$ as follows:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1 & 0 & \cdots & 0 \\ 0 & \phi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \phi_S \end{bmatrix}. \tag{14}$$

Other VAR structures consider ϕ as a linear combination of spatial covariates and cross-correlation between time series (Santos-Fernandez et al., 2022). These variations are not currently implemented in **SSNbayes** but are under development.

2.5 Bayesian inference and specification of prior distributions

Formulating this model in a Bayesian framework requires sampling from the following posterior distribution:

$$[\boldsymbol{\beta}, \boldsymbol{\Phi}, \sigma_0^2, \sigma_u^2, \alpha_u, \sigma_d^2, \alpha_d, \sigma_e^2, \alpha_e \mid \mathbf{y}, \mathbf{X}]. \tag{15}$$

SSNbayes uses Stan (Carpenter et al., 2017) to efficiently produce samples from this distribution via Hamiltonian Monte Carlo (HMC) methods.

We also need to define prior distributions for the parameters of interest. Currently, diffuse prior distributions are the only option in `ssnbayes()`, but the functionality to include other prior distributions may be included in future package versions. The implemented prior distributions are the following:

$\phi \sim \text{Uniform}\,(-1,1)$	# uniform prior on the autoregressive parameter
$\alpha_{.} \sim \text{Uniform}\,(0, 4 \times \max(h))$	# diffuse prior on the spatial range
$\sigma_{.} \sim \text{Uniform}\,(0,100)$	# diffuse prior on the square root of the partial sill
$\sigma_0 \sim \text{Uniform}\,(0,100)$	# diffuse prior on the square root of the nugget effect
$\beta \sim \mathcal{N}\,(0,1000)$	# prior on the regression coefficients (intercept and slope)

For the autoregressive parameter (ϕ), a uniform prior defined from -1 to 1 is used to ensure the process is stationary. The upper limit for the spatial range parameter is set to four times the maximum stream distance between observation locations on the network.

2.6 Prediction

There are two ways of making predictions in **SSNbayes**. By default, predictions are produced for missing values (NA) in the response variable in the observation dataset used to fit the model via the posterior predictive distribution:

$$p(\hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{X}, \hat{\mathbf{X}}) = \int p(\hat{\mathbf{y}} \mid \eta, \hat{\mathbf{X}}) p(\eta \mid \mathbf{y}, \mathbf{X})\, \mathrm{d}\eta, \tag{16}$$

where $\hat{\mathbf{X}}$ are the covariates for $\hat{\mathbf{y}}$. This Eq 16 gives the probability distribution of a new observation, $\hat{\mathbf{y}}$, given the observed data $\mathbf{y}$ and covariates/predicted covariates.

However, this approach is not recommended when making predictions at a large number of spatial locations (e.g. predicting over an extensive branching stream network) since it would involve operations with large matrices which can be very inefficient.

The second approach uses the fitted model produced using `ssnbayes()` to generate predictions using the Kriging predictor in a prediction dataset (Banerjee et al., 2014; Gelfand et al., 2019). This produces estimates as a weighted average of observations.

$$\widehat{\boldsymbol{y}}_P = \boldsymbol{X}_P\boldsymbol{\beta} + \boldsymbol{C}_{OP}{}'\boldsymbol{C}_{OO}^{-1}(\boldsymbol{y}_O - \boldsymbol{X}_O\boldsymbol{\beta}), \tag{17}$$

where subscripts O and P indicate the observation and prediction locations, respectively. The stacked vector $\widehat{\boldsymbol{y}}_P$ contains the predictions at the P spatial locations across all the time points T. The observations are represented in the stacked vector $\boldsymbol{y}_O$, which contains all of the observations across the T time points. $\boldsymbol{X}_P$ and $\boldsymbol{X}_O$ are space-time design matrices of covariates for the observations and predictions, respectively, while $\boldsymbol{\beta}$ is a vector of regression coefficients.

The separable square matrix $\boldsymbol{C}_{OO}$ of dimension $n = S \times T$ (defined in Eq 1), contains the covariances between observations at all time points, where O and T are the number of observations and time points respectively. Similarly, $\boldsymbol{C}_{OP}$ is a $O \times T$ by $P \times T$ rectangular separable matrix of covariances between observation and prediction locations at all time points with the same structure as $\boldsymbol{C}_{OO}$. That is, if $\boldsymbol{C}_{OO}$ was obtained from an AR exponential tail-down model with parameters ϕ, σ_{td} and α_{td}, these same parameters are used to construct $\boldsymbol{C}_{OP}$.

The covariance matrix of observations ($\boldsymbol{C}_{OO}$) must be inverted at each MCMC iteration when making predictions (Eq. 17) and this quickly becomes computationally challenging for large datasets. However, the separability of (13) significantly reduces the computational burden in these cases (Wikle et al., 2019)

$$C_{OO}^{-1} = \boldsymbol{\Sigma}_{ar1}^{-1} \otimes \boldsymbol{\Sigma}_{OO}^{-1},$$

where $\mathbf{\Sigma}_{OO}$ is the spatial covariance matrix defined in Eq 8, and $\mathbf{\Sigma}_{ar1}$ is the temporal covariance matrix of an AR(1) process (13) which has an analytical inverse that is tridiagonal.

3 The SSNbayes package

3.1 Data pre-processing

Detailed spatial, topological, and attribute data are needed to fit spatial stream network models, including those implemented in **SSNbayes**. There are currently two software packages that can be used to generate this information: 1) the Spatial Tools for the Analysis of River Systems (STARS) custom toolset (Peterson and Ver Hoef, 2014) for ArcGIS version $\geq$ 10.1 (ESRI, 2019) or 2) the R package **openSTARS** (Kattwinkel et al., 2020). When the pre-processing is complete, both tools create a new directory with the extension `.ssn` (i.e. a `.ssn` object), which contains shapefiles of the stream network, observed locations, and prediction locations (optional). It also contains the response, covariates (optional), and the information needed to generate stream distances and spatial weights between observed and prediction locations. Several unique identifiers are also assigned to observed and prediction locations to denote unique locations (`locID`) and unique measurements in space and time (`pid`). If real data are not being used, the `createSSN()` and `SimulateOnSSN()` functions found in **SSN** can be used to generate artificial .ssn objects that meet these requirements. It is important to note, however, that the **SSN** package has been archived on CRAN. In the absence of a .ssn object, **SSNbayes** can still be used to fit models based solely on Euclidean covariance models.

3.2 Installation

The **SSNbayes** package for R statistical software ($\geq$ 3.3.0) extends the models implemented in the **SSN2** package. **SSNbayes** is based on the R package **rstan** and the C++ toolchain is required. See `https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started`.

The **SSNbayes** package can be found at CRAN and Github (`https://github.com/EdgarSantos-Fernandez/SSNbayes`) and installed using:

```
if(!require('SSNbayes')) install.packages("SSNbayes", dependencies = T)
```

Or:

```
remotes::install_github("EdgarSantos-Fernandez/SSNbayes", dependencies = T)
```

For this demonstration we will need the companion `R` package **SSNdata** (Santos-Fernandez, 2022).

```
remotes::install_github("EdgarSantos-Fernandez/SSNdata",
    ref = "HEAD", upgrade_dependencies = F, dependencies = F)
```

SSNbayes describes both spatial and temporal autocorrelation using Bayesian inference. Table 1 shows a summary of the main functions included in the package and a description of the most important arguments. The function `collapse()` is used to extract line features from an spatial stream network (ssn) object in a format suitable for visualisation using packages such as **ggplot2**. The function `ssnbayes()` is the core function in **SSNbayes**, providing the functionality to fit linear spatio-temporal regression models (Santos-Fernandez et al., 2022), while the `predict()` function is used to perform spatio-temporal prediction from a `stanfit` object generated from `ssnbayes()`. Generic functions such as `summary()` and `plot` can be used to generate summary statistics and visualize model outcomes. In addition, the helper functions `dist_weight_mat()` and `dist_weight_mat_preds()` provide the functionality to extract a list of Euclidean and stream distance matrices, a spatial weight matrix, and an indicator matrix for flow connectivity between sites.

3.3 Modelling stream temperatures

We use the dataset described in the Introduction to illustrate how the **SSNbayes** package can be used to explore, analyse and draw conclusions from a Bayesian spatio-temporal model. The `.ssn` object for these data is part of the **SSNdata** package.

```
if(!require('SSNdata')) remotes::install_github("EdgarSantos-Fernandez/SSNdata")
```

Table 1: A description of functions in the **SSNbayes** package and the most important arguments.

function	arguments	description
`collapse()`	t	Path to a `.ssn` object
`ssnbayes()`	formula	A formula object, as used in lm()
	data	A long data.frame containing the locations, dates, covariates and response
	path	Path with the name of the `.ssn` object
	space_method	A list defining whether a spatial stream network (ssn) object is used and the spatial corr
	time_method	A list specifying the temporal structure
	iter	Number of iterations
	warmup	Number of warm up samples
	chains	Number of chains
	addfunccol	(optional) Column name for site additive function
`predict()`	formula	A formula object, as used in lm()
	obs_data	A long data.frame containing the locations, dates, covariates and response
	stanfit	A stanfit object returned from `ssnbayes()`
	pred_data	A long df of predictions with the locations, dates, covariates and the response
	nsamples	The number of samples to draw from the posterior distributions
	addfunccol	(optional) Column name for site additive function
	locID_pred	(optional) the location id for the predictions
	chunk_size	(optional) the number of `locID` to make prediction from
`dist_weight_mat()`	t	Path to a `.ssn` object
`dist_weight_mat_preds()`	t	Path to a `.ssn` object

For reproducibility, we also created a Kaggle notebook containing the example from this section (`https://www.kaggle.com/edsans/ssnbayes`). For completeness, similar analyses were performed on simulated data and the results are presented in the Appendix.

The `.ssn` object was generated using the STARS custom toolset (Peterson and Ver Hoef, 2014) and contains 18 observation and 60 prediction locations spaced at 1km intervals along the stream network. Hourly temperature recordings were taken at the observation sites but these were averaged to mean daily values for the two-year period of the data set. The data residing within the `.ssn` object are imported into R, converted to an ssn object and pair-wise distances are calculated for all observed sites, observed and prediction sites, and prediction sites using the following **SSN2** package functions:

```
path <- system.file("extdata/clearwater.ssn", package = "SSNdata")
n <-  SSN2::ssn_import(path, predpts = "preds", overwrite  = TRUE)
SSN2::ssn_create_distmat(n,
                            predpts = "preds" ,
                            overwrite = TRUE,
                            among_predpts  = TRUE)
```

We also read in a data frame containing the response and covariate data:

```
clear <- readRDS(system.file("extdata/clear_obs.RDS", package = "SSNdata"))
```

In the data.frame `clear`, the response variable (temp) is the mean daily stream temperature measured at 18 observation sites. Here we focus on a subsample of longitudinal response data consisting of 24 time points at those sites over two years (Figure 3). We randomly split the dataset, with 2/3 used for training the model and 1/3 for testing the out-of-sample prediction accuracy. This training/testing split was performed once for illustration purposes but for complex datasets we recommend using leave-one-out cross-validation.

Stream temperature is strongly influenced by topography and climate variables (Isaak et al., 2017). The following covariates were available for the observation and prediction locations across all the time points: stream slope, elevation, watershed area (Isaak et al., 2017), and air temperature (e.g. Bal et al., 2014). In addition, we included the first pair of harmonic covariates (Fourier terms) for the time periods ($\sin_t$ and $\cos_t$) (Hyndman and Khandakar, 2008).

3.4 Visualizing stream network data in space and time

We begin by extracting the streams from the ssn object so that we can visualise the data in **ggplot2**.

```
n.df <- SSNbayes::collapse(n)
```

The data.frame n.df contains data describing the spatial location of individual stream segments, along with the additive function column. The spatial and space-time data can then be visualised using **ggplot2** (Fig 1).

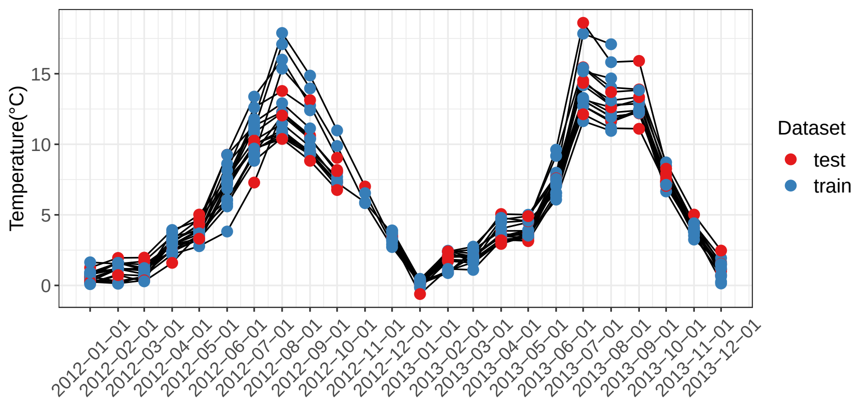

Figure 3: Time series of stream temperatures at multiple locations. Each line represents the time series for a unique observation location. The x-axis represents date, while the y-axis represents the water temperature in C°. Observations (training dataset) are displayed as blue points and the predictions (testing set) are displayed in red. The plot reveals the periodic changes in water temperatures over time at different locations in the stream network.

3.5 Fitting spatio-temporal linear models

The next step is to fit a linear spatio-temporal regression model using the function `ssnbayes()`. We specify the following linear regression model using the covariates in the observed dataset:

$$X'_{(t)}\beta = \beta_0 + \beta_1 * \text{SLOPE} + \beta_2 * \text{elev} + \beta_3 * \text{h2o\_area} + \beta_4 * \text{air\_temp}_{(t)} + \beta_5 * \sin_{(t)} + \beta_6 * \cos_{(t)}, \quad (18)$$

and fit the model to the observed temperature data using the `ssnbayes()` function.

```
fit_ar <- ssnbayes(formula = temp ~ SLOPE + elev + h2o_area + air_temp + sin + cos,
                   data = clear,
                   path = path,
                   space_method = list("use_ssn", "Exponential.taildown"),
                   time_method = list("ar", "date"),
                   iter = 3000,
                   warmup = 1500,
                   chains = 3,
                   net = 2,
                   addfunccol='afvArea',
                   refresh = max(iter/100,1))
```

Running this function takes several minutes and the progress of the sampler is shown during the execution. We stored the fitted model within **SSNdata**, which can be accessed using the code below if the reader wants to skip fitting the model.

```
fit_ar <- readRDS(system.file("extdata//fit_ar.rds", package = "SSNdata"))
```

The reader is referred to the Appendix for a second reproducible example using simulated data.

In the `ssnbayes()` function call shown above, the argument `formula` describes the regression model and is defined in the same way as in other model-fitting functions such as `lm`. We also pass a data.frame using the `data` argument, which must contain all of the variables specified in the `formula` argument. This data.frame should be in long format, with one row for each unique observation in space and time, which are also defined using `locID` and `pid`. In addition, the data set needs to be structured such that each spatial location has the same number of temporal observations. Such observations can be missing but should be denoted as such via "NA". In such cases, Bayesian imputation is used to obtain a complete data set. In other words, the data.frame has to contain all the combinations S and T. This can be done e.g. using the `tidyr::complete()` function.

The `space_method` argument is a list containing information about the spatial modelling component. The first element specifies whether the topological information is stored in a ssn object or not ("use_ssn" or "no_ssn"), while the second list element specifies which spatial correlation model(s) to use. Options include tail-up ("Exponential.tailup", "LinearSill.tailup", "Spherical.tailup"), tail-down ("Exponential.taildown", "LinearSill.taildown", "Spherical.taildown") and Euclidean ("Exponential.Euclid") models. It is possible to have more than one spatial covariance function per family (tail-up, tail-down and Euclidean distance). For instance: `space_method = list('use_ssn', c("Exponential.tailup", "Spherical.taildown"))`. However, care should be taken in this case to ensure identifiability of the model. If the user specifies use_ssn as the first element and the second element in the list is missing, then an "Exponential.tailup" model will be used by default. When a tail-up covariance function is specified, an additional column containing the additive function values used to compute the spatial weights must also be specified (e.g. `addfunccol` ='afvArea').

The argument `net` specifies the network identifier when multiple networks are found within the same ssn object. Much less information is needed to fit traditional Euclidean covariance models and so a ssn object is not needed. Instead, the columns containing the spatial coordinates (e.g. longitude and latitude) must be included as a third element in the list: `space_method = list("no_ssn", "Exponential.Euclid", c("lon", "lat"))`.

The temporal part of the model is defined in a similar fashion using a list `time_method = list("method", "date")`. The first element defines the temporal model and options include an autoregressive model, "ar", defined in Eq 12 or a vector autoregression model, "var", defined in Eq 14). The second element is the variable defining the time points in the observation data.frame, which must be a discrete numeric variable. They should also be spaced at regular intervals, as expected in many time series models.

In **SSNbayes** the number of chains (`chains`), iterations (`iter`), and burn-in samples (`warmup`) can be specified. By default, `chains` = 3, `iter` = 3000, `warmup` = 1500. Thinning is also possible using the argument `thin`. Optionally, the `seed` parameter can be set to ensure reproducibility.

The **SSNbayes** package depends on **rstan**, which does not allow missing values in the data. Missing values in the response variable (left hand side element in the formula argument) will be automatically imputed in the `ssnbayes()` function. However, missing values in the covariates (right hand side elements in the formula argument) are not allowed. Instead, they must be imputed by the user before fitting the model. Many options for imputation can be found in https://cran.r-project.org/web/views/MissingData.html.

The `ssnbayes()` function shows the progress of the model fit and will be updated based on the number of samples specified using the `refresh` argument. At every iteration, the inverse of the spatial covariance matrix has to be computed, which takes a substantial amount of time for a large number of spatial locations and time points. Fitting this dataset using the `ssnbayes()` function took approximately 10 minutes on a laptop with an Intel Core i7-8650U CPU @ 1.90GHz and 16 Gb of memory.

3.6 Exploring results

The output from `ssnbayes()` is a `stanfit` object, which contains information about the fitted model and the MCMC chains for the parameters of interest. It can be summarized and visualized using generic functions such as `summary()` and `plot()`, or functions in the **ggplot2** package. We can also visualize the posterior distributions in the parameters of interest using the `mcmc_dens_overlay()` function from the R package **bayesplot** (Gabry and Mahr, 2018). The regression coefficients across three chains are shown in Figure 4).

```
library('bayesplot')
mcmc_dens_overlay(
  fit_ar,
  pars = paste0("beta[",1:7,"]"),
  facet_args = list(nrow = 1))
```

Apart from h2o_area (β_3), all of the estimated regression coefficients for covariates are substantially different from zero. The bulk of the posterior distribution of the autoregressive parameter (ϕ) was also far away from zero, suggesting a strong temporal dependence (Figure 5).

The `mcmc_dens_overlay()` function can also be used to visualise the posterior distributions of the spatial model parameters (σ^2_{TD} and α_{TD}) and the nugget effect (σ^2_0) (Figure 6).

```
mcmc_dens_overlay(
  fit_ar,
  pars = c("var_td", "alpha_td", "var_nug"),
  facet_args = list(nrow = 1))
```

Notice that the median of the spatial range α_{TU} is approximately 200,000 m, indicating that spatial autocorrelation exists between locations that are less than 200 km apart.

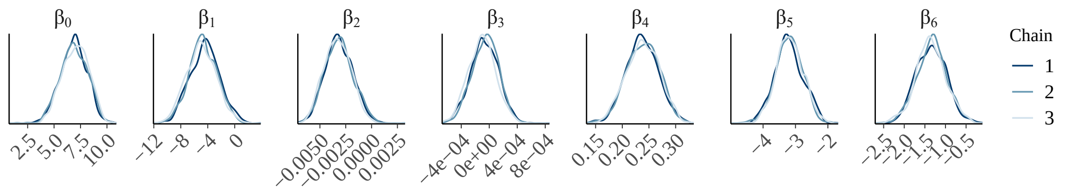

Figure 4: Posterior distributions of the regression coefficients for the linear model with seven covariates, including intercept (β_0), stream slope (β_1), elevation (β_2), watershed area (β_3), air temperature (β_4), sin of day of year (β_5), and cos of day of year (β_6). The distributions are shown for chains 1, 2, and 3, providing insights into the uncertainty and variability of the model coefficients.

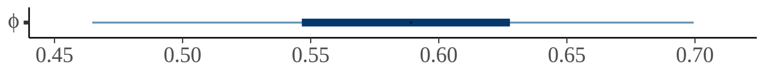

Figure 5: Boxplot of the posterior distribution for the autoregression parameter, ϕ, estimated from the stream temperature time series data. The box represents the interquartile range (IQR). The distribution shows moderate uncertainty in estimating the value of ϕ.

3.7 Predictions

Ecological and environmental monitoring on stream networks generally produces data at discrete locations, which represent only a small section of the stream network. However, it is often desirable to make predictions in areas where data have not been collected to create spatially continuous maps (Isaak et al., 2017). In this section, we illustrate (1) how the model imputes missing values producing predictions, and (2) how to use the fitted model to predict in unsampled locations using a Kriging approach.

As mentioned previously, the `ssnbayes()` function imputes missing values in the response variable. The time series corresponding to the 18 spatial locations are shown in Fig 7. The model captures the periodic patterns in stream temperatures well, even in locations where most of the observations were missing (e.g. 8 and 12, Fig 7). We also compared the predictions produced by the model with the true latent hold-out data (Fig 8). If the model predictions were perfect we would expect points to fall on the diagonal line. The results suggest that the Bayesian model produces predictions that are similar to the true latent values. Most of the predictions (96%) were included within the 95% highest density interval, showing appropriate coverage of the predictions. The root mean square prediction error (RMSPE) between the true temperature values and the predictions was 0.510 °C, which is small considering the magnitude of variation in temperature values.

Additionally, in our case study, we want to produce temperature predictions at 60 locations generated using a systematic design (≈1km apart). The function `predict()` produces predictions using information contained in the `stanfit` object obtained from `ssnbayes()`. The argument `nsamples` specifies the number of random samples to select from the posterior distributions and it must be smaller than or equal to the number of iterations `iter` specified in `ssnbayes()`.

```
# reading the prediction data
clear_preds <- readRDS(system.file("extdata/clear_preds.RDS", package = "SSNdata"))
pred <- predict(path = path,
                           obs_data = clear,
                           stanfit = fit_ar,
```

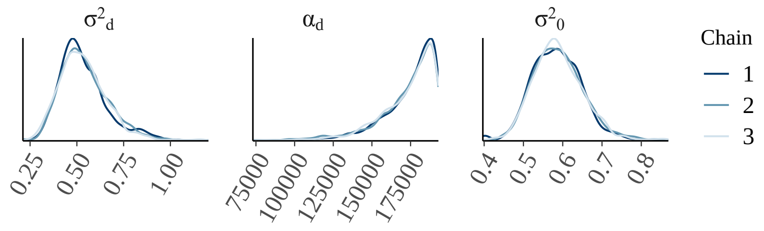

Figure 6: Posterior distributions of the spatial parameters, including the nugget effect (σ_0^2) and the spatial dependence (σ_d^2) in C°, and the range parameter (α_d) in meters.

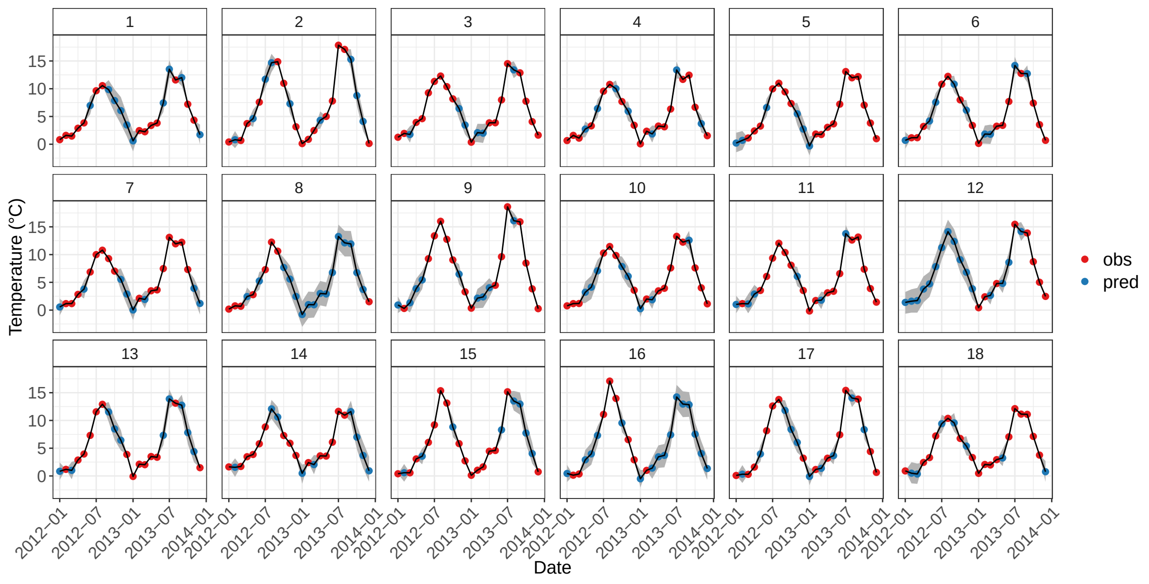

Figure 7: Time series of stream temperatures at 18 spatial locations. The observed points are represented in red, and the imputed or predicted values are shown in blue. The gray areas represent the 95% posterior credible intervals, providing an indication of the model's uncertainty.

```
pred_data = clear_preds,
net = 2,
nsamples = 100, # number of samples to use from the posterior
addfunccol = 'afvArea', # additive function values
locID_pred = locID_pred,
chunk_size = 60)
```

The observation and prediction data frames (data_obs, data_pred, respectively) must be specified and must contain all of the covariates and the response variable specified in the "formula" argument in ssnbayes().

Generally, producing subsets of predictions on the stream network is more efficient for big datasets, and can be parallelized. The argument chunk_size is used to define the size of the subsets. For instance, predictions in the case study in Santos-Fernandez et al. (2022) consist of more than 6000 locations. Performing matrix operations with a such large number of sites is not feasible. Instead, predictions were run in parallel using chunk_size = 100. locID_pred also allows the user to define a subset of prediction locations where predictions should be generated, as demonstrated in the example below. Similarly, the argument seed allows the user to set a seed so that the results are reproducible.

Figure 9 shows the predicted time series for the observation and prediction locations. The patterns in the prediction time series captured the seasonality in the observed data well. Figure 10 visualizes the predictions' posterior mean temperature on the stream network. As expected, higher temperature values are obtained in the main stream channel, compared to predictions in small streams which generally are found at higher elevations.

Network exceedance probability

One advantage of using Bayesian inference is the ability to easily obtain various probabilistic estimates based on the model posterior predictive samples. In this example, we use the function melt from the R package **reshape2** (Wickham, 2007) to generate exceedance probabilities based on a critical thermal threshold of 13 °C for bull trout, a cold-water fish species that is sensitive to increased temperatures.

```
ys <- reshape2::melt(pred, id.vars = c('locID0', 'locID', 'date'), value.name ='y')
ys$iter <- gsub("[^0-9.-]", "", ys$variable)
ys$variable <- NULL
# network exceedance probability
limit <- 13
ys$exc <- ifelse(ys$y > limit , 1, 0)
ys <- data.frame(ys) %>% dplyr::group_by(date, locID, locID0) %>%
```

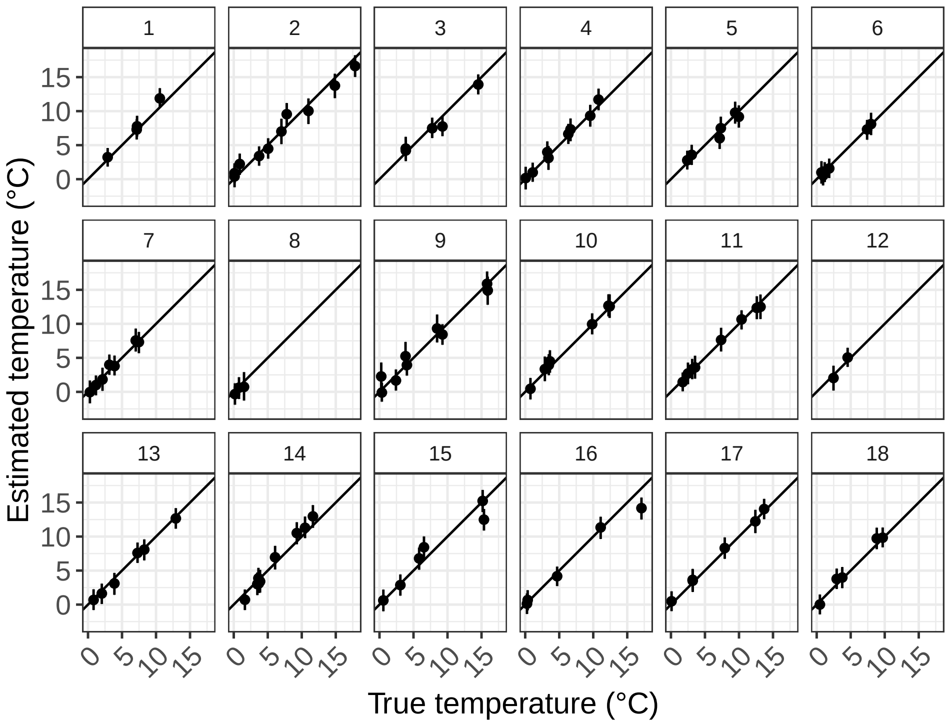

Figure 8: Scatterplot of predicted temperature versus the true latent values at the 18 spatial locations in the test dataset. The x and y axes represent the predicted and true temperature values, respectively. The vertical bars represent the 95% posterior credible intervals.

```
  dplyr::summarise(sd = sd(y, na.rm=T),
                   y_pred = mean(y, na.rm=T),
                   prop = mean(exc, na.rm=T)) %>%
  dplyr::arrange(ys, locID)
clear_preds <- clear_preds %>% left_join(ys, by = c('locID', 'date' ))
```

Figure 11 shows the exceedance probabilities for all 60 prediction locations on two dates, obtained from the posterior predictive distributions. Knowledge about when and where biologically relevant thermal thresholds are likely to be exceeded provides critical information for the management of threatened and endangered freshwater species (Isaak et al., 2016).

Other useful functions

Users often want to extract distance matrices and spatial weights so that they can be analysed and/or visualised in other R packages or external software e.g. McGuire et al. (2014). The `dist_weight_mat()` and `dist_weight_mat_preds()` produce a list of distance and weight matrices with the following elements:

1. e: Euclidean distance matrix containing the distances between locations
2. D: Downstream distance.
3. H: Total stream distance.
4. w.matrix: spatial weights for flow connected locations. This matrix is used in the tail-up models.
5. flow.con.mat: flow connected matrix. Indicates whether two locations in the network are connected by flow.

The `dist_weight_mat()` function produces matrices of the distances and weights between observation locations, with dimensions equal to the number of observation locations ($n_o \times n_o$). The `dist_weight_mat_preds()` function produces the same information for observed and prediction locations, with $n_o n_p \times n_o n_p$ dimensions. Details and detailed descriptions of the computation of these matrices can be found in Peterson and Ver Hoef (2010), Ver Hoef et al. (2014), and Santos-Fernandez et al. (2022).

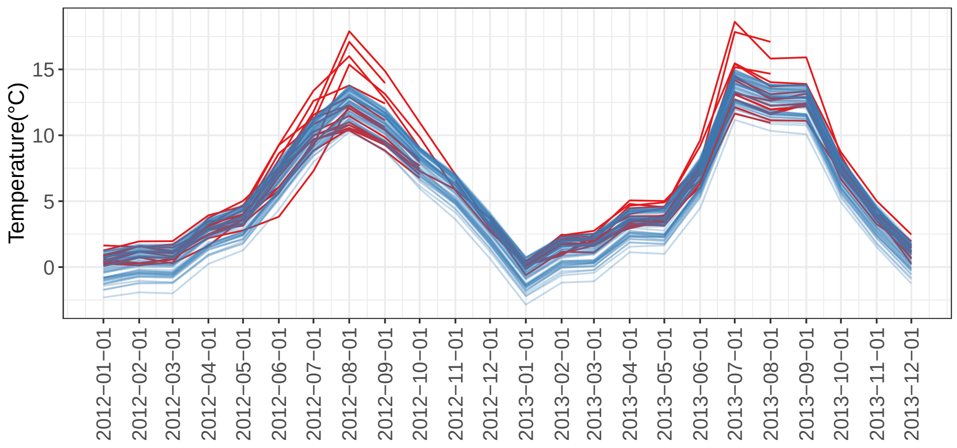

Figure 9: Time series plot comparing predicted (blue lines) and observed (red lines) temperature values over time.

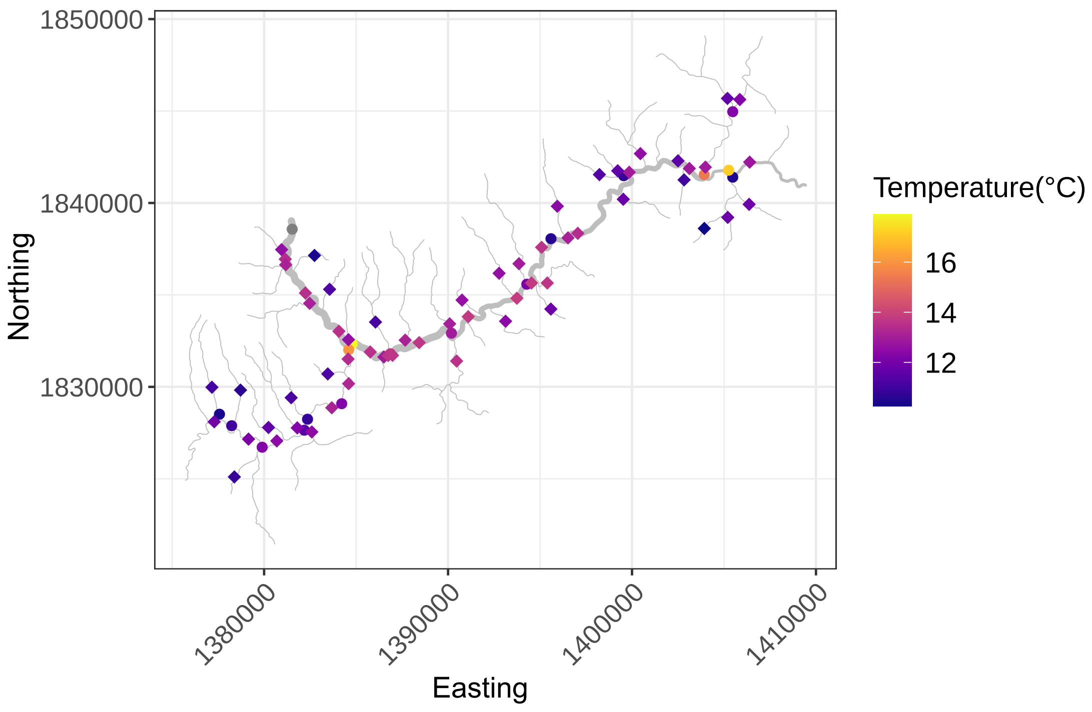

Figure 10: Mean daily stream temperature predictions (diamonds) for each of the 60 spatial locations and observations (circles) in the Clearwater network on August 1st, 2012.

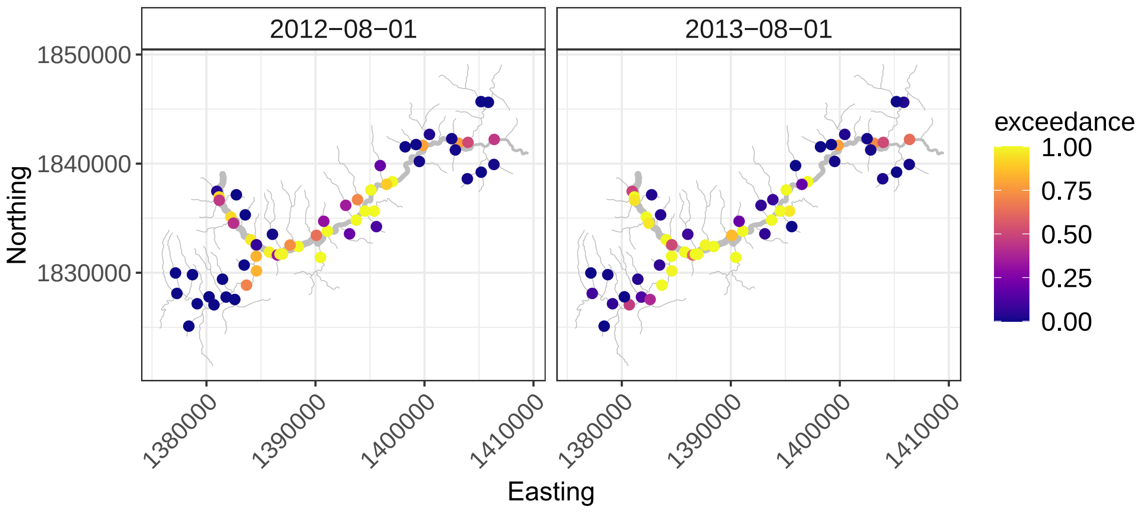

Figure 11: Probabilities that the mean stream temperature will exceed the 13 °C threshold on August 1, 2012, and August 1, 2013, respectively across the 60 prediction locations in the Clearwater network.

4 Discussion and conclusions

The growing popularity of stream sensor arrays, in which repeated observations are taken at multiple sites, requires models capable of accounting for spatial and temporal autocorrelation in stream network data. However, there are only a limited number of computational methods and software packages designed to account for the unique spatial dependence found in stream data (e.g. the R packages **SSN**, **SSN2** and **smnet**). The package described in the present paper extends the models implemented in **SSN** by accounting for temporal dependence using Bayesian inference, which offers several benefits. Enhanced features from this package and other benefits from the use of a Bayesian framework include the computation of probabilistic estimates and network exceedance probabilities, the ability to incorporate prior information, and the estimation of the proportion of degraded habitat.

We tested the performance of **SSNbayes** in multiple scenarios with simulated and real data and found that the parameters are well estimated and the predictions are accurate in terms of RMSPE. We also validated the results from a wide range of spatial model combinations to those obtained using **SSN** based on simulated data. Spatial and spatio-temporal models tend to be slow to fit and computationally intensive, which becomes more challenging within a Bayesian modelling framework. This can become computationally prohibitive when the number of spatial locations is large because the spatial covariance matrix must be iteratively inverted. We are currently exploring alternative methods (such as variational Bayes) to be implemented within **SSNbayes**.

Future implementations will incorporate other modelling variations. Two of them are: (I) expressing ϕ_s as a linear combination of covariates such as elevation and watershed (an extension of Case 2), and (II) using a 2-Nearest Neighbours (2-NN) method, where the off-diagonal elements of $\mathbf{\Phi}$ are different from zero in the two closest, allowing temporal dependence to be established between neighbouring spatial locations connected by flow (Santos-Fernandez et al., 2022). However, other space-time covariance structures could also be implemented for stream network data, which allow more modelling flexibility. For example, this implementation is based on a vector autoregression structure, but other models such as moving averages and ARIMA could also be considered. In addition, we currently assume that the response variable is normally distributed, but other regression models could be implemented by modifying the likelihood function in `ssnbayes()`. We are also actively working on the development and implementation of models for anomaly detection in stream data (Santos-Fernandez et al., 2023). The **SSNbayes** package is under constant development and new features and model implementations are on their way.

Acknowledgement

This research was supported by the Australian Research Council (ARC) Linkage Project "Revolutionising water-quality monitoring in the information age" (ID: LP180101151) and the Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS). JMM was supported by an Australian Research Council Discovery Project (DP200101263). We thank Dona Horan for the creation of the spatial stream network (.ssn) object. Data analysis and computations were undertaken using the packages **rstan** (Stan Development Team, 2018). Data visualizations were made with the packages **ggplot2** (Wickham, 2016) and **bayesplot** (Gabry and Mahr, 2018).

Appendix

In this appendix, we illustrate the fit of a spatio-temporal stream network model within the Bayesian framework with **SSNbayes** using a simulated example. These results can be reproduced using the Kaggle notebook `https://www.kaggle.com/code/edsans/ssnbayes-simulated`. Several R packages must be installed to successfully reproduce the simulation described in this section. Installing the **rstan** can be tricky because you need to configure your R installation to be able to compile C++ code. If you have not used the **rstan** before, please see https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started for easy-to-follow instructions about how to install the package.

Using simulated data

We generated some spatial data with the **SSN** package using a systematic design for the locations of the observations and predictions. Note that the **SSN** package has been archived on CRAN.

```
## Load the packages. Note that these packages can be installed
## using the install.packages() function
library('tidyverse')
library('Rcpp')
library('StanHeaders')
library('rstan')
library('abind')
library('SSN')
library('SSN2')
library('bayesplot')
library('lubridate')
library('viridis')
library('ggrepel')
library('devtools')
library('RColorBrewer')
if(!require('SSNbayes')) install.packages("SSNbayes", dependencies = T)
library('SSNbayes')

## Set some useful options for modelling
rstan_options(auto_write = TRUE) # avoid recompilation
options(mc.cores = parallel::detectCores())
RNGkind(sample.kind = "Rounding")

## Set the seed for reproducibility
seed <- 202008
set.seed(seed)

## Set the path for the SpatialStreamNetwork object created in the next step
path <- "./sim.ssn"

## If it does not already exist, create a SpatialStreamNetworkObject
## with 150 stream segments (edges). Use a systematic design to generate
## observed and prediction locations on the network spaced
## approximately 3 and 0.3 units apart, respectively.
if(file.exists(path)){
  ssn <- importSSN(path, "preds")
}  else{ssn <- createSSN(n = c(150),  # 150 edges
               obsDesign = systematicDesign(spacing=3),
               predDesign = systematicDesign(spacing=0.3),
               importToR = TRUE,
               path = path, # path where the sns object is saved
               treeFunction = iterativeTreeLayout)}
```

This produces a SpatialStreamNetwork object with 50 observation locations, which is our training dataset. We also generated 499 prediction locations for testing.

```
## Plot the edges and observed locations in the SpatialStreamNetwork object
plot(ssn, lwdLineCol = "addfunccol",  lwdLineEx = 8,
     lineCol = 4,  col = 1,  pch = 16,  xlab = "x-coordinate",  ylab = "y-coordinate")

## Add the prediction locations
plot(ssn, PredPointsID = "preds", add = T, pch = 16, col = "#E41A1C")
```

Consider a response variable such as stream temperature. The aim is to predict this response variable in the testing dataset borrowing information from other measurements across space and time and using covariates such as air temperature. Predictions can be made at a subset of locations or across all prediction locations on the whole network.

Distances within and between observation and prediction locations must be generated before model fitting.

```
## Create stream distance matrices
createDistMat(ssn, predpts = 'preds',o.write=TRUE, amongpreds = T)
```

We then need to simulate some data using some covariates, regression coefficients and a covariance structure.

```
## Extract the data.frames for the observed and prediction location data
rawDFobs <- getSSNdata.frame(ssn, Name = "Obs")
rawDFpred <- getSSNdata.frame(ssn, Name = "preds")

## Extract the geographic coordinates from the SpatialStreamNetwork
## object and add to data.frames
obs_data_coord <- data.frame(ssn@obspoints@SSNPoints[[1]]@point.coords)
obs_data_coord$pid<- as.numeric(rownames(obs_data_coord))
rawDFobs<- rawDFobs %>% left_join(obs_data_coord, by = c("pid"),
                                  keep = FALSE)
rawDFobs$point <- "Obs" ## Create label for observed points

pred_data_coord <- data.frame(ssn@predpoints@SSNPoints[[1]]@point.coords)
pred_data_coord$pid<- as.numeric(rownames(pred_data_coord))
rawDFpred<- rawDFpred %>% left_join(pred_data_coord, by = "pid",
                                    keep = FALSE)
rawDFpred$point <- "pred" ## Create label for prediction points

## Generate 3 continuous covariates at observed and prediction locations
set.seed(seed)
rawDFpred[,"X1"] <- rnorm(length(rawDFpred[,1]))
rawDFpred[,"X2"] <- rnorm(length(rawDFpred[,1]))
rawDFpred[,"X3"] <- rnorm(length(rawDFpred[,1]))

rawDFobs[,"X1"] <- rnorm(length(rawDFobs[,1]))
rawDFobs[,"X2"] <- rnorm(length(rawDFobs[,1]))
rawDFobs[,"X3"] <- rnorm(length(rawDFobs[,1]))

## Ensure the rownames still match the pid values used in the
## SpatialStreamNetwork object
rownames(rawDFobs)<- as.character(rawDFobs$pid)
rownames(rawDFpred)<- as.character(rawDFpred$pid)

## Put the new covariates back in the SpatialStreamNetwork object
ssn <- putSSNdata.frame(rawDFobs,ssn, Name = 'Obs')
ssn <- putSSNdata.frame(rawDFpred, ssn , Name = 'preds')
```

```
## Simulate the response variable at observed and prediction locations
set.seed(seed)
sim.out <- SimulateOnSSN(ssn.object = ssn,
                          ObsSimDF = rawDFobs, ## observed data.frame
                          PredSimDF = rawDFpred, ## prediction data.frame
                          PredID = "preds", ## name of prediction dataset
                          formula = ~ X1 + X2 + X3,
                          coefficients = c(10, 1, 0, -1), ## regression coefficients
                          CorModels = c("Exponential.taildown"), ## covariance model
                          use.nugget = TRUE, ## include nugget effect
                          CorParms = c(3, 10, .1)) ## covariance parameters

## Extract the SpatialStreamNetwork object from the list returned by
## SimulateOnSSN and extract the observed and prediction site
## data.frames. Notice the new column Sim_Values in the data.frames
sim.ssn <- sim.out$ssn.object
simDFobs <- getSSNdata.frame(sim.ssn,"Obs")
simDFpreds <- getSSNdata.frame(sim.ssn, "preds")
summary(simDFobs)
```

The SpatialStreamNetwork object we created only contains one simulated response for each observation and prediction location and we can fit a spatial statistical model to the simulated data using the glmssn function in the **SSN** package.

```
## Fit a spatial stream network model using the Exponential tail-down function
glmssn.out <- glmssn(Sim_Values ~ X1 + X2 + X3, sim.ssn,
                     CorModels = "Exponential.taildown")
summary(glmssn.out)
```

In order to fit a space-time model using the **SSNbayes** package, we need repeated measurements at each location. We now need to generate some time series with AR(1) error structure:

```
## Create a data.frame containing training and test data.
df_obs <- getSSNdata.frame(sim.ssn, "Obs") ## Extract observed dataset
df_obs$dataset <- 'train' ## Create new column 'dataset' and set to 'train'
df_pred <- getSSNdata.frame(sim.ssn, "preds") ## Extract prediction dataset
df_pred$dataset <- 'test' ## Create new column 'dataset' and set to 'test'

## Expand data.frames to include 10 days per location
t <- 10 # days
df_obs <- do.call("rbind", replicate(t, df_obs, simplify = FALSE))# replicating the df
df_obs$date <- rep(1:t, each = (nrow(df_obs)/t)) # Set date variable

df_pred <- do.call("rbind", replicate(t, df_pred, simplify = FALSE))# replicating the df
df_pred$date <- rep(1:t, each = (nrow(df_pred)/t)) # Set date variable

## Create a copy of the pid value used in the SpatialStreamNetwork
## object and create a new pid value for use in SSNbayes
## package. Values must be consequtively ordered from 1 to the number
## of rows in the data.frame
df_obs <- df_obs %>% mutate(pid.ssn = pid,
                            pid = rep(1:nrow(.)))
df_pred <- df_pred %>% mutate(pid.ssn = pid,
                              pid = rep(1:nrow(.)))

## Combine the training and testing datasets
df <- rbind(df_obs, df_pred)
df$dataset <- factor(df$dataset, levels = c('train', 'test'))

## Construct and initialize an autocorrelation structure of order 1
set.seed(seed)
phi <- 0.8 ## lag 1 autocorrelation value
ar1 <- corAR1(form = ~ unique(df$date), value = phi) # can also use corExp function
AR1 <- Initialize(ar1, data = data.frame(unique(df$date)))
```

```
## Create a vector of AR1 errors for each date and expand to all all locations
epsilon <- t(chol(corMatrix(AR1))) %*% rnorm(length(unique(df$date)), 0, 3)   #NB AR1 error
epsilon <- rep(epsilon, each = length(unique(df$locID)) ) +
  rnorm(length(epsilon)*length(unique(df$locID)), 0, 0.25) # for all the locations
epsilon_df <- data.frame(date = rep(unique(df$date), each = length(unique(df$locID))),
                         locID = rep(unique(df$locID), times = length(unique(df$date))),
                         epsilon = epsilon)
df <- df %>% left_join(epsilon_df, by = c('date' = 'date', 'locID' = 'locID'))

## Create a new simulated response variable, y, with errors added
df$y <- df$Sim_Values + df$epsilon
```

We can visualize the time series of the new simulated response at the observed and predicted locations over time:

```
## Create line plots of response over time for training and test datasets
ggplot(df) +
  geom_line(aes(x = date, y = y, group = locID, col = dataset), alpha = 0.4) +
  ylab("Simulated Temperature (\u00B0C)")+
  facet_wrap(~dataset)+
  theme_bw()
```

Figure 12 shows the stream temperature time series in the observations and predictions datasets.

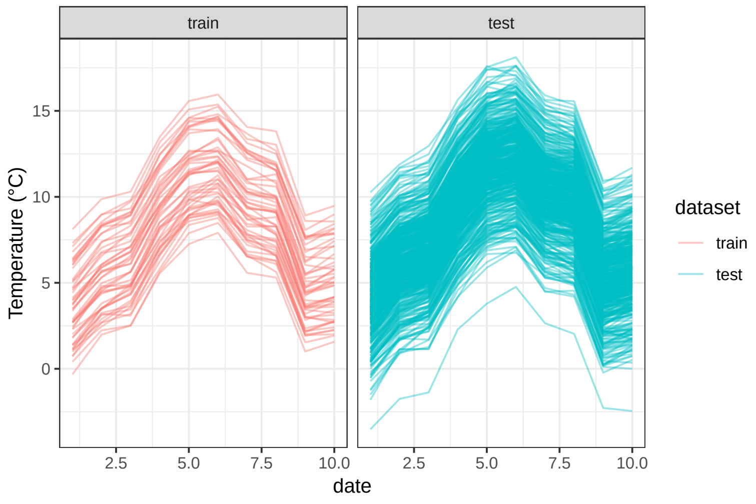

Figure 12: Evolution of the stream temperature time series in the observations and predictions datasets. Each time series represent a spatial location.

```
## Split the training and testing datasets. Ensure that date is numeric.
df <- df %>% dplyr::select(locID, pid, date, y, everything())
obs_data <- df[df$dataset == 'train',]
pred_data <- df[df$dataset == 'test',]
# NB: the order in this data.fame MUST be: spatial locations (1 to S) at time t=1,
# then locations (1 to S) at t=2 and so on.
```

The first prediction option in SSNbayes is to generate predictions for locations in the observed dataset with missing data. To demonstrate, let us set approximately 30% of the observations per date to missing, make predictions, and assess how well we retrieve the actual temperature values.

```
# Randomly select observations by date
set.seed(seed)
points <- length(unique(obs_data$pid))
locs <- obs_data %>% dplyr::group_by(date) %>%
   pull(pid) %>%
  sample(., round(points * 0.3), replace = F)  %>% sort()

## Create a backup for the response before setting randomly selected
## measurements to NA
obs_data$y_backup <- obs_data$y
obs_data[obs_data$pid %in% locs,]$y <- NA
```

Let us visualize the network with the time series of observed temperature values. First we use the collapse() function to extract the network structure from the the SpatialStreamNetwork object. The facets (1-10) represent the date and there are a total of 150 missing observations (gray dots) which are observations that we set to missing to assess the model predictive accuracy.

```
## Extract stream (edge) network structure, including the additive function value
nets <- SSNbayes::collapse(ssn, par = 'addfunccol')

## Create additive function value categories for plotting
## Create additive function value categories for plotting
nets$afv_cat <- cut(nets$addfunccol,
                                 breaks = seq(min(nets$addfunccol),
                                              max(nets$addfunccol),
                                              length.out=6),
                                 labels = 1:5,
                                 include.lowest = T)

## Plot simulated temperature, by date, with line width proportional to afv_cat
ggplot(nets) +
    geom_path(aes(X1, X2, group = slot, size = afv_cat), lineend = 'round',
              linejoin = 'round', col = 'lightblue')+
    geom_point(data = dplyr::filter(obs_data, date %in% 1:10),
               aes(x = coords.x1, y = coords.x2, col = y, shape = point),
               size = 1)+
  scale_size_manual(values = seq(0.2,2,length.out = 5))+
  facet_wrap(~date, nrow = 2)+
  scale_color_viridis(option = 'C')+
  scale_shape_manual(values = c(16,15))+
  xlab("x-coordinate") +
  ylab("y-coordinate")+
  theme_bw()
```

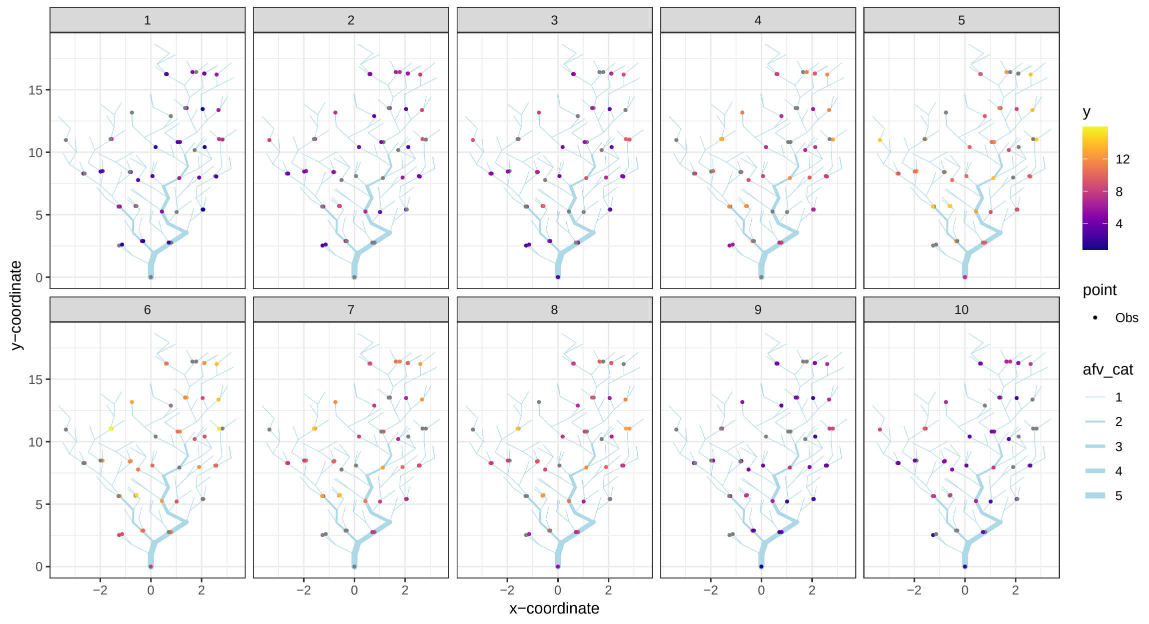

Figure 13: Evolution of the stream temperature time series in the observation dataset across ten time points.

We then fit a Bayesian space-time model, with a tail-down covariance model and an AR(1) error structure using the ssnbayes() function. Fitting the model took about 15 minutes on a laptop (i7 @1.80 GHz and 16GB RAM).

```
## Fit a Bayesian space-time model
fit_td <- ssnbayes(formula = y ~ X1 + X2 + X3,
                   data = obs_data,
                   path = path,
                   time_method = list("ar", "date"), # temporal model to use
                   space_method = list('use_ssn', c("Exponential.taildown")), # spatial model
                   iter = 4000,
                   warmup = 2000,
                   chains = 3,
                   addfunccol = 'addfunccol',
                   loglik = T)

## Create a copy of the model fit and set class so that we can take
## advantage of plotting functions for stanfit objects
fits <- fit_td
class(fits) <- c("stanfit")

## Extract summaries of the posterior distributions for the parameter
## estimates and the predictions.
stats_td <- summary(fits)
stats_td <- stats_td$summary
```

One of the main benefits of this Bayesian approach is that the model produces probabilistic estimates. Figures 14 and 15 show the posterior distributions of the four parameters in the spatio-temporal model (σ^2_{TU}, σ^2_0, α and ϕ) and the regression coefficients (intercept and slopes). The trace plots for of these parameters can be found in the Appendix .0.1.

```
## Create plots of the posterior distribution of the 3 regression
## coefficients
mcmc_dens_overlay(
  fits, #
  pars = paste0("beta[",1:3,"]"),
  facet_args = list(nrow = 1))

## Plot the posterior distribution of phi
mcmc_intervals(
  fits,
  pars = paste0("phi"),
  point_size = .1,
  prob_outer = 0.95
)

## Plot the posterior distribution for the nugget effect, partial sill.
## and range parameters in the tail-down model
mcmc_dens_overlay(
  fits,
  pars = c(
    "var_td",
    "alpha_td",
    "var_nug"),
  facet_args = list(nrow = 1)
)
```

We then assess how accurate the predictions of the missing temperature values are compared to the true held out values.

```
## Create a data.frame containing summaries of the posterior predictive
## distributions and the true values
ypred <- data.frame(stats_td[grep("y\\[", row.names(stats_td)),])
ypred$ytrue <- obs_data$y_backup #
ypred$date <- rep(1:t, each = nrow(obs_data)/t)
```

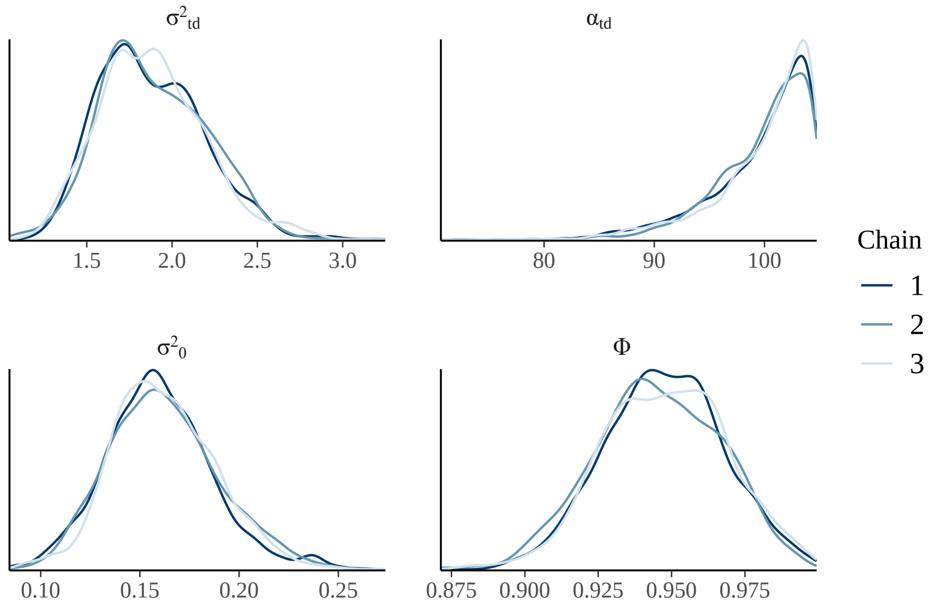

Figure 14: Posterior densities of the partial sill (σ^2_{TU}), nugget effect (σ^2_0), range (α) and temporal autocorrelation ϕ parameters. These parameters are crucial for understanding the spatial and temporal variability in the simulated stream network dataset.

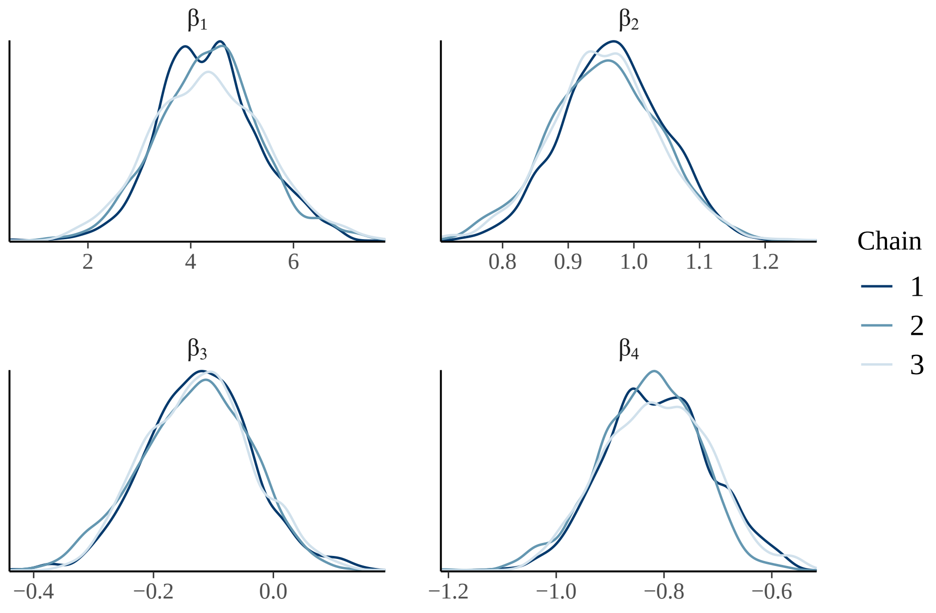

Figure 15: Posterior densities of the regression coefficients (β) for the intercept (β_1) and the three covariates X1, X2, and X3 (β_2, β_3, and β_4, respectively).

```
ypred$dataset <- ifelse(ypred$sd == 0, 'obs', 'pred')
ypred$td_exp <- ypred$mean

## Create a plot of the predicted versus true values with 95% highest
## density interval
 filter(ypred, dataset == 'pred') %>% ggplot() +
     geom_errorbar(data = ypred, aes(x=ytrue, ymin=X97.5., ymax=X2.5.),
                   col = 2, width=0.5, size=0.5, alpha = 0.75) +
  geom_point(aes(x = ytrue , y = td_exp), col = 2)+
  geom_abline(intercept = 0, slope = 1)+
  facet_wrap(~date, nrow = 2) +  coord_fixed() +
     xlab('y true') + ylab('y estimated') +  theme_bw()

## Calculate the root mean square error (RMSE) between the true and
## predicted values
rmse <- sqrt(mean(((ypred$ytrue) - ypred$td_exp)^2))
rmse

# Calculate the 95% prediction coverage. Ideally, this should be close to 0.95.
ypred$cov <- ifelse(ypred$ytrue > ypred$X2.5. & ypred$ytrue<ypred$X97.5,1,0)
filter(ypred, dataset == 'pred') %>%
  group_by(dataset) %>%
  dplyr::summarize(mean(cov))
```

In Figure 17, we have plotted the means of the imputed values (with a 95% prediction interval) against the true values of the missing data. As can be seen, the means of each imputed value largely agree with the true value, with most prediction intervals (96%) containing the true value. The RMSE of the predicted temperature (y) is 0.245 which is small compared to the magnitude of y (Mean = 7.621 and standard deviation = 3.489 degrees). Figure 16 shows the posterior densities of the predicted temperature (y) for 8 measurements in the testing set.

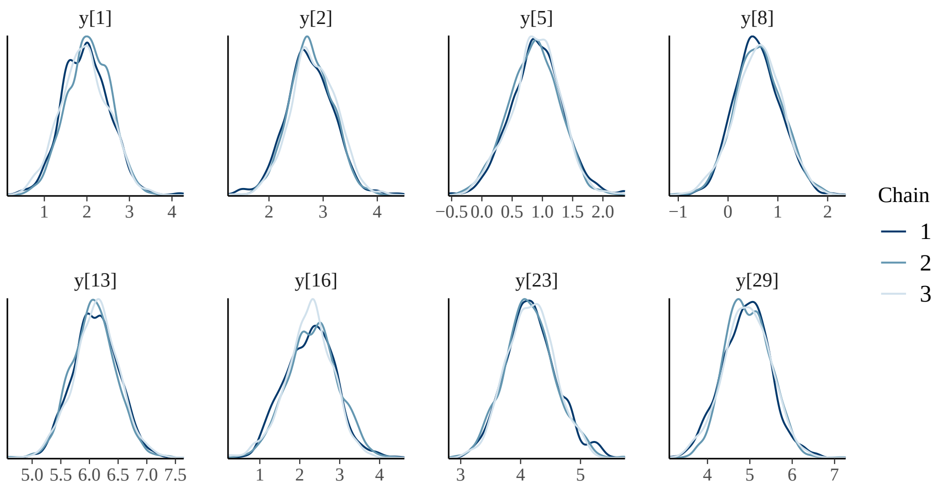

Figure 16: Posterior densities for 8 temperature predictions (y).

0.1 Predictions on a new set of prediction locations

Environmental and ecological monitoring of stream networks often requires estimation of the response variable of interest across the whole river network. In this section we illustrate how to use the fitted model to predict at a new set of locations using Kriging. We will produce temperature predictions at the 499 locations we generated previously using a systematic design.

```
## Making predictions at a new set of prediction locations

## Set the seed for reproducibility
```

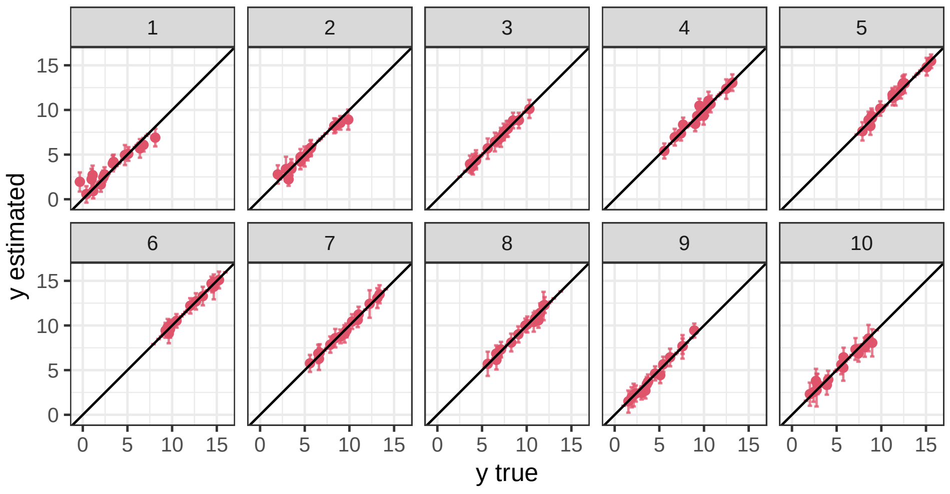

Figure 17: Scatterplot of predicted (i.e., estimated) temperature values versus the true latent values. The error bars represent the 95% highest density interval, indicating the range of the most plausible true values given the predictions.

```
set.seed(seed)

## Extract the location IDs for prediction locations
locID_pred <- sort(unique(pred_data$locID))

## Define a new column containing the response variable used in
## fit_td, including missing values
obs_data$y_traintest <- obs_data$y

## Replace y with the original response variable, containing no
## missing values
obs_data$y <- obs_data$y_backup

## Produce predictions: This takes approximately 8 minutes
pred <- predict(object = fit_td, ## fitted model
                path = path, #  path to .ssn object
                obs_data = obs_data, # observed data.frame
                pred_data = pred_data, # prediction data.frame
                net = 1, # network identifier (optional)
                nsamples = 100, # number of samples to use from the posterior
                addfunccol = 'addfunccol', # variable used for spatial weights
                locID_pred = locID_pred, # location identifier for predictions
                chunk_size = 60) # split the predictions into subsets of this size

## Convert the prediction data.frame from wide to long format
ys <- reshape2::melt(pred, id.vars = c('locID0',
                                       'locID', 'date'), value.name ='y')

## Create variable representing the iteration number and set variable
## column to NULL
ys$iter <- gsub("[^0-9.-]", "", ys$variable)
ys$variable <- NULL
```

As we are using a simulated dataset, we can compare the out-of-sample predictions with the true latent values in the same way we did previously (Figure 18).

We then visualize the posterior mean of the predictions on the full network (Figure 19).

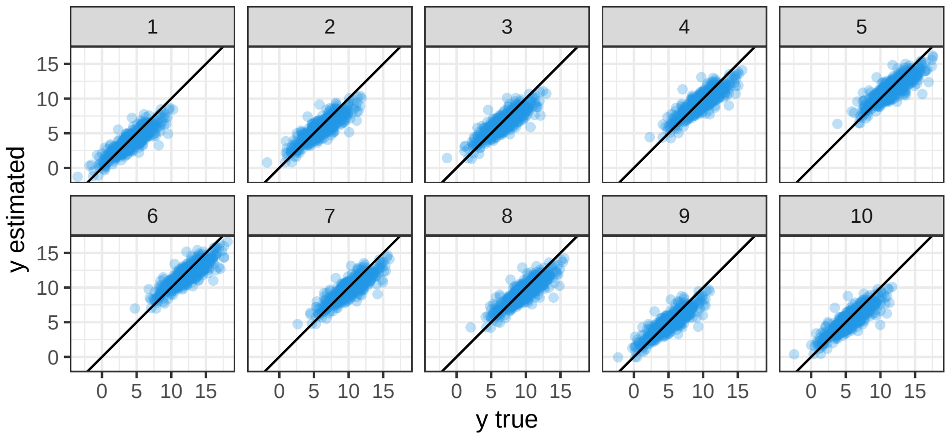

Figure 18: Scatter plot of predicted (i.e. estimated) temperature versus the true latent values. The plot shows a strong linear relationship between the two, indicating good agreement between the true and estimated values.

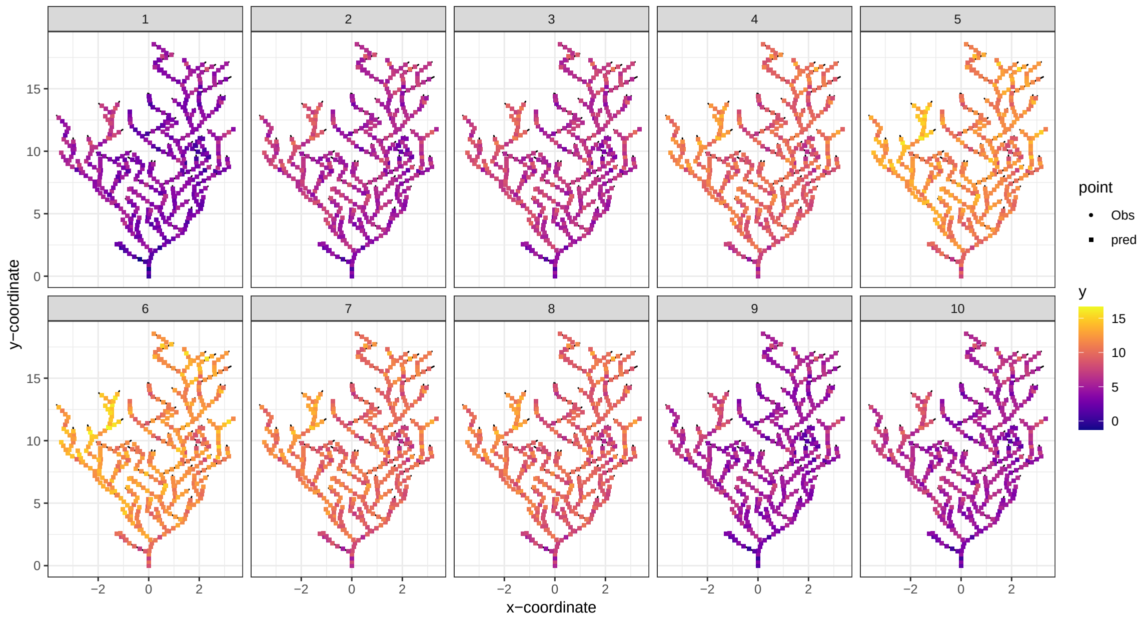

Figure 19: Evolution of observed and predicted stream temperatures over the 10 dates. The plot highlights the variability in water temperatures over time.

Exceedance probabilities throughout the network

It is straightforward to make probabilistic statements about the predictions based on samples from the posterior predictive distribution. In this example, we calculate the exceedance probabilities at each prediction location based on a biological threshold of 13 °C.

```
## Computing the exceedance probabilities

## Create an exceedance indicator based on a threshold (i.e. limit)
limit <- 13
ys$exc <- ifelse(ys$y > limit , 1, 0) ## 1== TRUE, 0== FALSE

## Calculate summary statistics for the predictions, by locID and date
## and join to prediction data.frame
ys <- data.frame(ys) %>% dplyr::group_by(date, locID) %>%
  dplyr::summarise(sd = sd(y, na.rm=T), ## prediction standard deviation
                   y_pred = mean(y, na.rm=T), ## mean temperature prediction
                   prop = mean(exc, na.rm=T)) ## exceedance probability

ys <- dplyr::arrange(ys, locID)
pred_data <- pred_data %>% left_join(ys, by = c('locID', 'date'), keep = FALSE)


## Plot the exceedance probabilities for the prediction sites on 10 dates
ggplot(nets) +
  geom_path(aes(X1, X2, group = slot, size = afv_cat), lineend = 'round',
            linejoin = 'round', col = 'gray')+
  geom_point(data = dplyr::filter(pred_data) ,
             aes(x = coords.x1, y = coords.x2, col = prop), size = 1)+
  scale_size_manual(values = seq(0.2,2,length.out = 5))+
  facet_wrap(~date, nrow = 2)+
  scale_color_viridis(option = 'C')+
  scale_shape_manual(values = c(200))+
  xlab("x-coordinate") +
  ylab("y-coordinate")+
  coord_fixed()+
  theme_bw()+
  theme(axis.text=element_text(size=12),
        axis.title=element_text(size=13),
        legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        strip.text.x = element_text(size = 13),
        axis.text.x = element_text(angle = 45, hjust=1),
        strip.background =element_rect(fill='white'))+
  guides(size = 'none')+
    labs(size="", colour="exceedance")
```

Figure 20 shows the time series of the exceedance probabilities at prediction locations obtained from the posterior predictive distributions. Knowing when and where temperature or other water quality variables are likely to exceed critical thresholds provides valuable information for prioritizing management and conservation activities.

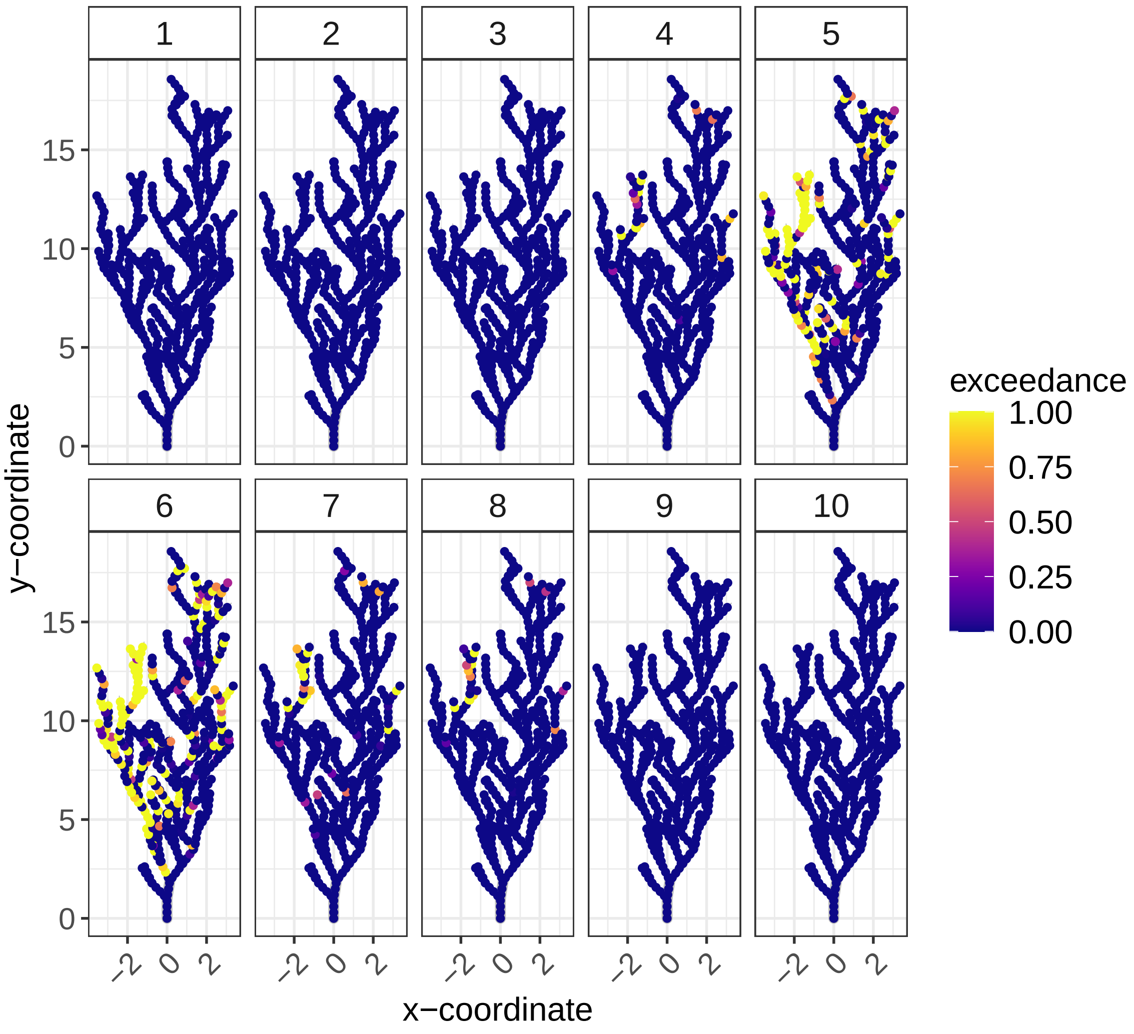

Figure 20: Network exceedance probabilities across the ten dates. The yellow regions represent the areas where the probability of exceeding the limit is high.

Other results

Trace plots are often used to visually assess convergence when parameters are estimated in a Bayesian model. These can be generated easily from the `stanfit` object we saved earlier (fits) and the `traceplot` function in **rstan** .

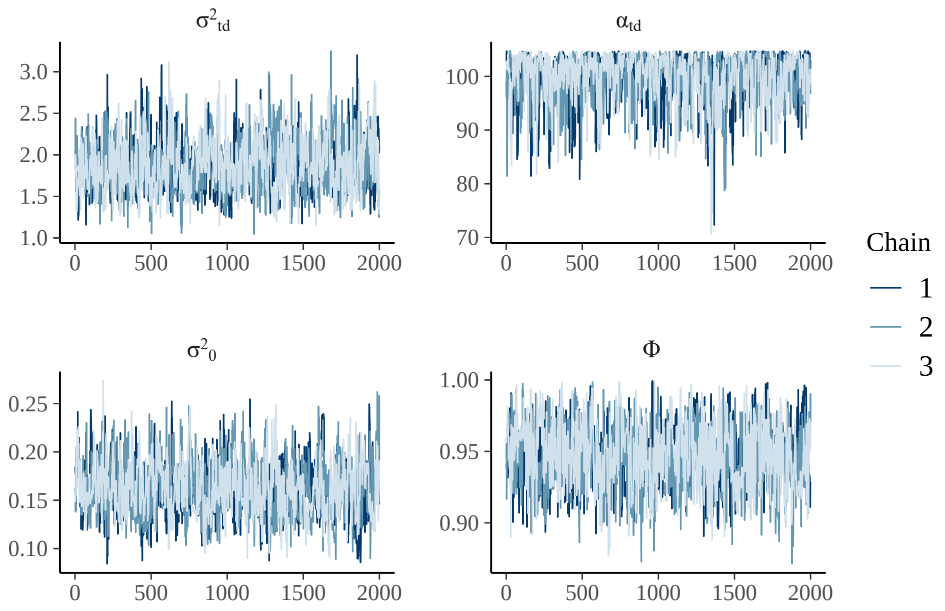

Figure 21: Trace plots of the spatial covariance parameters including the partial sill (σ^2_{TD}), nugget effect (σ^2_0), range (α), and the temporal autocorrelation parameter, ϕ, showing good mixing of the chains.

References

G. Bal, E. Rivot, J.-L. Baglinière, J. White, and E. Prévost. A hierarchical Bayesian model to quantify uncertainty of stream water temperature forecasts. *PLoS One*, 9(12):e115659, 2014. [p34]

S. Banerjee, B. P. Carlin, and A. E. Gelfand. *Hierarchical modeling and analysis for spatial data*. CRC press, 2014. [p28, 32]

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76 (1), 2017. [p27, 32]

N. Cressie and C. K. Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015. [p28]

N. Cressie, J. Frey, B. Harch, and M. Smith. Spatial prediction on a river network. *Journal of Agricultural, Biological, and Environmental Statistics*, 11(2):127, 2006. [p29]

C. Donegan. *geostan: Bayesian spatial analysis (R package)*, 2022. URL `https://cran.r-project.org/package=geostan`. The Comprehensive R Archive Network. [p27]

M. Dumelle, E. Peterson, J. M. Ver Hoef, A. Pearse, and D. Isaak. *SSN2: Spatial Modeling on Stream Networks in R*, 2023. R package version 0.1.0. [p26]

ESRI. *ArcGIS Desktop*. Environmental Systems Research Institute., Redlands, CA., 2019. [p33]

A. O. Finley, S. Banerjee, and A. E.Gelfand. spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63(13):1–28, 2015. URL `http://www.jstatsoft.org/v63/i13/`. [p27]

J. C. Frieden, E. E. Peterson, J. A. Webb, and P. M. Negus. Improving the predictive power of spatial statistical models of stream macroinvertebrates using weighted autocovariance functions. *Environmental Modelling & Software*, 60:320–330, 2014. [p30]

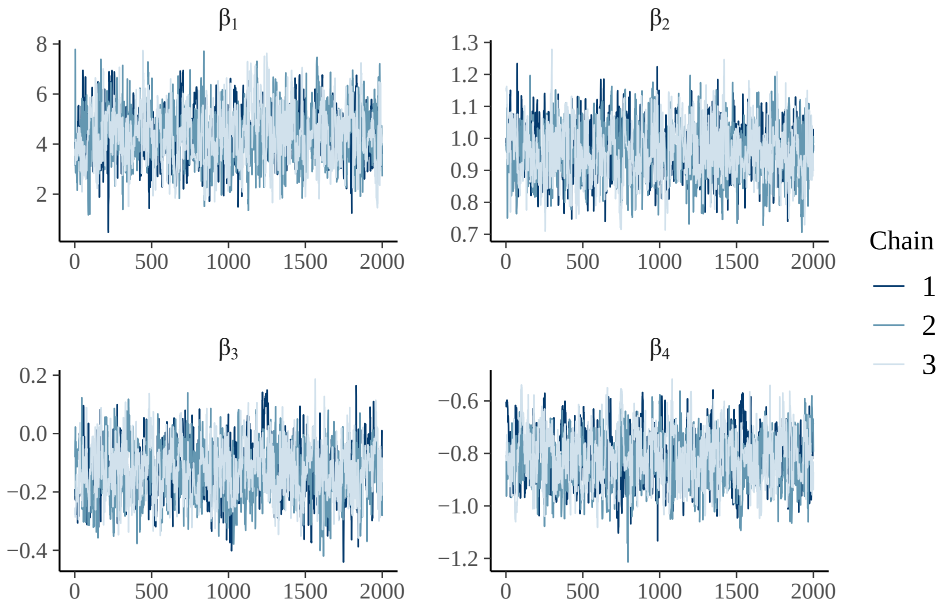

Figure 22: Trace plots of the regression coefficients (β).

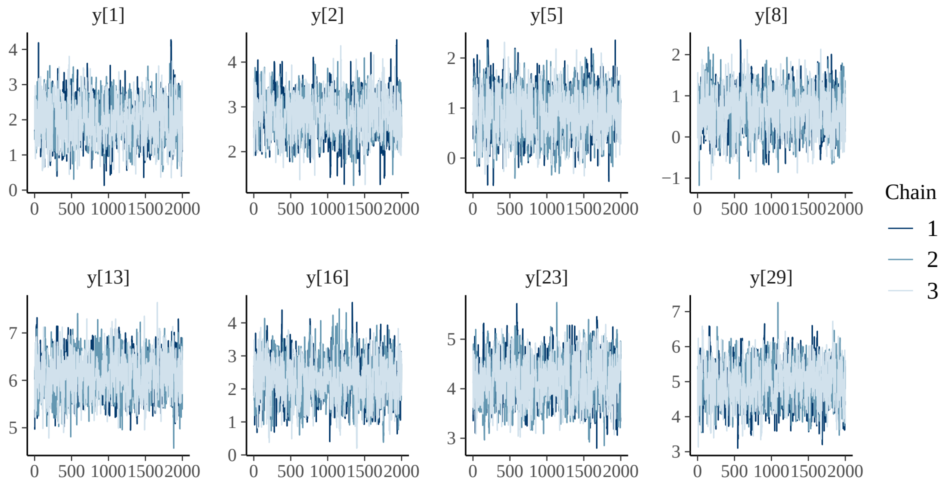

Figure 23: Trace plots of the predicted temperature (y) in eight prediction points.

J. Gabry and T. Mahr. *bayesplot: Plotting for Bayesian Models*, 2018. URL https://CRAN.R-project.org/package=bayesplot. R package version 1.6.0. [p36, 42]

V. Garreta, P. Monestiez, and J. M. Ver Hoef. Spatial modelling and prediction on river networks: up model, down model or hybrid? *Environmetrics*, 21(5):439–456, 2010. [p29]

A. E. Gelfand, M. Fuentes, J. A. Hoeting, and R. L. Smith. *Handbook of environmental and ecological statistics*. CRC Press, 2019. [p32]

J. D. Hamilton. *Time series analysis*. Princeton university press, 1994. [p31]

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. URL https://www.jstatsoft.org/article/view/v027i03. [p34]

D. J. Isaak, E. E. Peterson, J. M. Ver Hoef, S. J. Wenger, J. A. Falke, C. E. Torgersen, C. Sowder, E. A. Steel, M.-J. Fortin, C. E. Jordan, et al. Applications of spatial statistical network models to stream data. *Wiley Interdisciplinary Reviews: Water*, 1(3):277–294, 2014. [p29]

D. J. Isaak, M. K. Young, C. H. Luce, S. W. Hostetler, S. J. Wenger, E. E. Peterson, J. M. Ver Hoef, M. C. Groce, D. L. Horan, and D. E. Nagel. Slow climate velocities of mountain streams portend their role as refugia for cold-water biodiversity. *Proceedings of the National Academy of Sciences*, 113(16): 4374–4379, 2016. [p39]

D. J. Isaak, S. J. Wenger, E. E. Peterson, J. M. Ver Hoef, D. E. Nagel, C. H. Luce, S. W. Hostetler, J. B. Dunham, B. B. Roper, S. P. Wollrab, G. L. Chandler, D. L. Horan, and S. Parkes-Payne. The NorWeST summer stream temperature model and scenarios for the western US: A crowd-sourced database and new geospatial tools foster a user community and predict broad climate warming of rivers and streams. *Water Resources Research*, 53(11):9181–9205, 2017. doi: https://doi.org/10.1002/2017WR020969. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017WR020969. [p34, 37]

D. J. Isaak, C. H. Luce, G. L. Chandler, D. L. Horan, and S. P. Wollrab. Principal components of thermal regimes in mountain river networks. *Hydrology and Earth System Sciences*, 22(12):6225–6240, 2018. [p26]

F. L. Jackson, R. J. Fryer, D. M. Hannah, C. P. Millar, and I. A. Malcolm. A spatio-temporal statistical model of maximum daily river temperatures to inform the management of Scotland's Atlantic salmon rivers under climate change. *Science of the Total Environment*, 612:1543–1558, 2018. [p29]

M. Kattwinkel, E. Szöcs, E. Peterson, and R. B. Schäfer. Preparing gis data for analysis of stream monitoring data: The r package openstars. *PLOS ONE*, 15(9):1–10, 09 2020. doi: 10.1371/journal.pone.0239237. URL https://doi.org/10.1371/journal.pone.0239237. [p33]

D. Lee. CARBayes: An R package for Bayesian spatial modeling with conditional autoregressive priors. *Journal of Statistical Software*, 55(13):1–24, 2013. URL http://www.jstatsoft.org/v55/i13/. [p27]

F. Lindgren and H. Rue. Bayesian spatial modelling with R-INLA. *Journal of Statistical Software*, 63(19): 1–25, 2015. URL http://www.jstatsoft.org/v63/i19/. [p27]

K. J. McGuire, C. E. Torgersen, G. E. Likens, D. C. Buso, W. H. Lowe, and S. W. Bailey. Network analysis reveals multiscale controls on streamwater chemistry. *Proceedings of the National Academy of Sciences*, 111(19):7030–7035, 2014. doi: 10.1073/pnas.1404820111. URL https://www.pnas.org/doi/abs/10.1073/pnas.1404820111. [p39]

M. G. McManus, E. D'Amico, E. M. Smith, R. Polinsky, J. Ackerman, and K. Tyler. Variation in stream network relationships and geospatial predictions of watershed conductivity. *Freshwater Science*, 39 (4):704–721, 2020. [p29]

E. Money, G. P. Carter, and M. L. Serre. Using river distances in the space/time estimation of dissolved oxygen along two impaired river networks in new jersey. *Water Research*, 43(7):1948–1958, 2009a. [p29]

E. S. Money, G. P. Carter, and M. L. Serre. Modern space/time geostatistics using river distances: data integration of turbidity and E. coli measurements to assess fecal contamination along the raritan river in New Jersey. *Environmental Science & Technology*, 43(10):3736–3742, 2009b. [p29]

E. Pebesma. CRAN Task View: Handling and Analyzing Spatio-Temporal Data. https://cran.r-project.org/web/views/SpatioTemporal.html, 2021. Accessed: 2021-04-13. [p27]

E. Peterson and J. M. Ver Hoef. STARS: An arcgis toolset used to calculate the spatial information needed to fit spatial statistical models to stream network data. *Journal of Statistical Software*, 56(2): 1–17, 2014. [p30, 33, 34]

E. E. Peterson and J. M. Ver Hoef. A mixed-model moving-average approach to geostatistical modeling in stream networks. *Ecology*, 91(3):644–651, 2010. [p28, 30, 39]

E. E. Peterson, A. A. Merton, D. M. Theobald, and N. S. Urquhart. Patterns of spatial autocorrelation in stream water chemistry. *Environmental Monitoring and Assessment*, 121(1):571–596, 2006. [p28]

E. E. Peterson, J. M. Ver Hoef, D. J. Isaak, J. A. Falke, M.-J. Fortin, C. E. Jordan, K. McNyset, P. Monestiez, A. S. Ruesch, A. Sengupta, et al. Modelling dendritic ecological networks in space: an integrated network perspective. *Ecology Letters*, 16(5):707–719, 2013. [p28]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2020. URL `https://CRAN.R-project.org/package=nlme`. R package version 3.1-148. [p27]

J. C. Pinheiro and D. M. Bates. *Mixed-Effects Models in S and S-PLUS*. Springer, New York, 2000. doi: 10.1007/b98882. [p27]

P. J. Ribeiro Jr, P. J. Diggle, M. Schlather, R. Bivand, and B. Ripley. *geoR: Analysis of Geostatistical Data*, 2020. URL `https://CRAN.R-project.org/package=geoR`. R package version 1.8-1. [p27]

P. M. Rodríguez-González, C. García, A. Albuquerque, T. Monteiro-Henriques, C. Faria, J. B. Guimarães, D. Mendonça, F. Simões, M. T. Ferreira, A. Mendes, et al. A spatial stream-network approach assists in managing the remnant genetic diversity of riparian forests. *Scientific Reports*, 9(1):1–10, 2019. [p29]

A. Rushworth. *smnet: Smoothing for Stream Network Data*, 2017. URL `https://CRAN.R-project.org/package=smnet`. R package version 2.1.1. [p26]

S. K. Sahu, D. P. Lee, and K. S. Bakar. *bmstdr: Bayesian Modeling of Spatio-Temporal Data with R*, 2022. URL `https://CRAN.R-project.org/package=bmstdr`. R package version 0.3.0. [p27]

E. Santos-Fernandez. *SSNdata: Spatial stream network datasets*, 2022. URL `https://github.com/EdgarSantos-Fernandez/SSNdata`. [p33]

E. Santos-Fernandez, J. M. Ver Hoef, E. E. Peterson, J. McGree, D. J. Isaak, and K. Mengersen. Bayesian spatio-temporal models for stream networks. *Computational Statistics & Data Analysis*, 170:107446, 2022. [p30, 31, 33, 38, 39, 41]

E. Santos-Fernandez, J. M. Ver Hoef, E. E. Peterson, J. McGree, C. A. Villa, C. Leigh, R. Turner, C. Roberts, and K. Mengersen. Unsupervised anomaly detection in spatio-temporal stream network sensor data. *preprint*, 2023. [p41]

M. Schlather, A. Malinowski, P. J. Menck, M. Oesting, and K. Strokorb. Analysis, simulation and prediction of multivariate random fields with package RandomFields. *Journal of Statistical Software*, 63(8):1–25, 2015. URL `http://www.jstatsoft.org/v63/i08/`. [p27]

J. O. Skoien, G. Bloschl, G. Laaha, E. Pebesma, J. Parajka, and A. Viglione. Rtop: An r package for interpolation of data with a variable spatial support, with an example from river networks. *Computers & Geosciences*, 2014. [p26]

Stan Development Team. RStan: the R interface to Stan, 2018. URL `http://mc-stan.org/`. R package version 2.18.2. [p42]

J. M. Ver Hoef and E. E. Peterson. A moving average approach for spatial statistical models of stream networks. *Journal of the American Statistical Association*, 105(489):6–18, 2010. [p29, 30]

J. M. Ver Hoef, E. Peterson, and D. Theobald. Spatial statistical models that use flow and stream distance. *Environmental and Ecological statistics*, 13(4):449–464, 2006. [p29, 30]

J. M. Ver Hoef, E. Peterson, D. Clifford, and R. Shah. SSN: An R package for spatial statistical modeling on stream networks. *Journal of Statistical Software*, 56(3):1–45, 2014. [p26, 30, 39]

C. J. Vörösmarty, P. B. McIntyre, M. O. Gessner, D. Dudgeon, A. Prusevich, P. Green, S. Glidden, S. E. Bunn, C. A. Sullivan, C. R. Liermann, et al. Global threats to human water security and river biodiversity. *Nature*, 467(7315):555–561, 2010. [p26]

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL `http://www.jstatsoft.org/v21/i12/`. [p38]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `https://ggplot2.tidyverse.org`. [p42]

C. K. Wikle, L. M. Berliner, and N. Cressie. Hierarchical bayesian space-time models. *Environmental and Ecological Statistics*, 5(2):117–154, 1998. [p31]

C. K. Wikle, A. Zammit-Mangion, and N. Cressie. *Spatio-temporal Statistics with R*. CRC Press, 2019. [p32]

A. Zammit-Mangion and N. Cressie. Frk: an r package for spatial and spatio-temporal prediction with large datasets. *Journal of Statistical Software*, 98(4):1–48, 2021. [p27]

Edgar Santos-Fernandez
School of Mathematical Sciences, Queensland University of Technology
Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS)
Y Block, Floor 8, Gardens Point Campus. GPO Box 2434. Brisbane, QLD 4001.
Australia
ORCiD: 0000-0001-5962-5417
`santosfe@qut.edu.au`

Jay M. Ver Hoef
Marine Mammal Laboratory, NOAA-NMFS Alaska Fisheries Science Center.
Seattle, WA and Fairbanks, AK, USA
ORCiD: 0000-0003-4302-6895
`jay.verhoef@noaa.gov`

James McGree
School of Mathematical Sciences, Queensland University of Technology

ORCiD: 0000-0003-2997-8929
`james.mcgree@qut.edu.au`

Daniel J. Isaak
Rocky Mountain Research Station, US Forest Service

ORCiD: 0000-0002-8137-325X
`Daniel.isaak@usda.gov`

Kerrie Mengersen
School of Mathematical Sciences, Queensland University of Technology
Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS)
Y Block, Floor 8, Gardens Point Campus. GPO Box 2434. Brisbane, QLD 4001.
Australia
ORCiD: 0000-0001-8625-9168
`k.mengersen@qut.edu.au`

Erin E. Peterson
EP Consulting
School of Mathematical Sciences, Queensland University of Technology
Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS)
Y Block, Floor 8, Gardens Point Campus. GPO Box 2434. Brisbane, QLD 4001.
Australia
ORCiD: 0000-0003-2992-0372
`erin@peterson-consulting.com`