spStack: Practical Bayesian Geostatistics Using Predictive Stacking in R

Soumyakanti Pan
University of California
Los Angeles

Sudipto Banerjee
University of California
Los Angeles

Abstract

Statistical modeling and analysis for spatially oriented point-referenced outcomes play a crucial role in diverse scientific applications such as earth and environmental sciences, ecology, epidemiology, and economics. With the advent of Markov chain Monte Carlo (MCMC) algorithms, Bayesian hierarchical models have gained massive popularity in analyzing such point-referenced or, geostatistical data. However, these models involve latent spatial processes characterized by spatial process parameters, which besides lacking substantive relevance in scientific contexts, are also weakly identified and hence, impedes convergence of MCMC algorithms. Thus, even for moderate sized datasets, the computation for MCMC becomes too onerous for practical use. In this article, we introduce the R package **spStack** that delivers fast Bayesian inference for a class of geostatistical models, where we obviate these issues by sampling from analytically available posterior distributions conditional upon candidate values of the spatial process parameters and, subsequently assimilate inference from these individual posterior distributions using Bayesian predictive stacking. Besides delivering competitive predictive performance as compared to fully Bayesian inference using MCMC, our proposed algorithm is executable in parallel, thus drastically improving runtime and elevating the utility of our software to a diverse group of practitioners with limited computational resources at their disposal.

Keywords: Bayesian inference, spatial modeling, geostatistics, predictive stacking, R.

1. Introduction

With the onset of the open-access era, there has been growing interest in the scientific community to study spatial and temporal variability in complex datasets. Rapid growth in Geographical Information Systems (GIS) and Global Positioning Systems (GPS) in the last few decades has led to government as well as private agencies to collect and store spatially referenced data with regulatory, monitoring and resource management objectives. Recent technological advances has empowered cheap and seamless integration of these systems into a wide array of devices leading to collection of information at increasingly high spatial resolutions. The sources of these datasets are remarkably diverse, e.g., field measurements of particulate matter for air quality assessments in public health research, computer models used to simulate complex physical processes in earth sciences, species sightings in ecology, real estate pricing and land valuation in economics etc. Thus, there has been a rise in demand among researchers in various disciplines to analyze such point-referenced (latitude-longitude, Easting-Northing, etc.) data to better understand different sources of variability in their model of interest. Since rigorous statistical analysis equips the practitioner with scientific evidence, based on which they take decisions with important economic, environmental, and public health implications, it is crucial to estimate inferential uncertainty. In this aspect, Bayesian hierarchical models have managed to procure extensive appeal due to its capability of performing the non-trivial task of delivering fully model-based probabilistic quantification of inferential uncertainty.

In order to address the challenges in analysis of spatial data, statistical modeling has seen several significant developments; see, for example, the books by Cressie (1993), Moller and Waagepetersen (2003), Wackernagel (2003), Banerjee, Carlin, and Gelfand (2014), Schabenberger and Gotway (2005), Diggle and Ribeiro (2007), Wikle and Cressie (2011), Chilès and Delfiner (2012) for a variety of methods and applications. In the literature, it is widely accepted that spatial dependence is effectively captured by hierarchical models which incorporate dependencies at multiple levels. Inference for such hierarchical models fall under the Bayesian paradigm of statistical inference (see, e.g., Carlin and Louis 2008; Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin 2013) where analysis is based on samples of model parameters from their joint posterior distribution. Computational advances with regard to Markov chain Monte Carlo (MCMC) algorithms have helped Bayesian hierarchical models gain massive popularity in a diverse array of fields (see, e.g., Gilks, Richardson, and Spiegelhalter 1995; Robert and Casella 2004) including spatial modeling (Banerjee *et al.* 2014). This is largely owed to automated implementation of MCMC algorithms introduced by the BUGS (Bayesian Inference Using Gibbs Sampling) project (Lunn, Spiegelhalter, Thomas, and Best 2009) through their Windows-only software **WinBUGS** which was succeeded by **OpenBUGS** (Thomas, O'Hara, Ligges, and Sturtz 2006), and more recently, **JAGS** (Just Another Gibbs Sampler) and NIMBLE (de Valpine, Turek, Paciorek, Anderson-Bergman, Temple Lang, and Bodik 2017). These software packages offer user-friendly interface for constructing tailored multilevel models and subsequently samples from the posterior distribution using Gibbs sampler. The stage is also shared by Stan (Stan Development Team 2024), which uses Hamiltonian Monte Carlo methods (Neal 2011) for sampling from the posterior distribution. These software platforms interface with the popular R statistical environment (R Core Team 2024), making them accessible to a wide community of users.

A salient feature of Bayesian hierarchical geostatistical models is the presence of a latent spatial process that specifies the probability law of the point-referenced measurements of the outcome at a finite set of locations. While the aforementioned software packages can be used to build such models, their scope is rather limited. First, for a fully Bayesian model with priors on the spatial process parameters, the MCMC algorithm is dominated by evaluation of the posterior distribution, or if necessary, its gradient, requiring repeated expensive matrix operations that are poorly implemented in these platforms. Second, the sampling strategies pursued by the aforementioned software platforms are intended to explore a very high-dimensional parameter space, since at every step, they update all model parameters that includes the latent spatial process. Third, the spatial variance (partial sill), spatial decay, spatial smoothness parameters and the nugget, if present, which we collectively refer to as "process parameters", are weakly identified (Zhang 2004). All these factors contribute towards poor mixing of the chain, hence, delayed convergence, and, prolonged execution times; thus, demonstrating how the aforementioned software packages are ill-suited for Bayesian geostatistical models. For example, the **GeoBUGS** module (Thomas, Best, Lunn, Arnold, and

Spiegelhalter 2014), that offers utilities for geostatistical modeling within **WinBUGS**, warns users that their implementation of such models can be very slow even for moderately sized datasets, and suggests to either use strong informative priors for the process parameters or fix them a priori.

It is, therefore, not unreasonable to pursue methods that will yield robust inference for the spatial process and deliver spatial predictions of the outcome at arbitrary points ("kriging") while circumventing inference on a few weakly identified parameters. Rather than devising computational algorithms to achieve improved convergence, we develop a framework to conduct Bayesian inference for fixed effects and the latent spatial process by entirely avoiding convergence issues for MCMC or other iterative algorithms for geostatistical data. Instead, we build a hierarchical model that yields analytically accessible posterior distributions for the fixed effects and the spatial random effects subject to fixing a few weakly identified process parameters and some hyperparameters so that we can draw exact posterior samples for any fixed values of these parameters. For Gaussian data, this falls under the conjugate Bayesian linear modeling framework. For non-Gaussian data, this is achieved by availing of recent results developed in Bradley and Clinch (2024) on a new class of analytically accessible generalized conjugate multivariate (GCM) distributions for spatial models by extending the Diaconis-Ylvisaker family of conjugate priors for exponential families (Diaconis and Ylvisaker 1979). Subsequently, we combine the inference by stacking the models on a grid of candidate values the fixed parameters, supplied by the user. Stacking (Wolpert 1992; Breiman 1996; Clyde and Iversen 2013) is a model averaging procedure that is widely used in machine learning and has been shown (see, e.g., Le and Clarke 2017; Yao, Vehtari, Simpson, and Gelman 2018; Yao, Vehtari, and Gelman 2022; Yao, Piršl, Vehtari, and Gelman 2021) to be an effective alternative to traditional Bayesian model averaging (Madigan, Raftery, Volinsky, and Hoeting 1996; Hoeting, Madigan, Raftery, and Volinsky 1999).

However, fitting a Bayesian hierarchical spatial model requires specialized competencies in probability theory and programming which may exceed the scope of scientific practitioners in different disciplines, and hence, requires automation. The *Geostatistics* section of the *Analysis of Spatial Data* (Bivand and Nowosad 2024) Comprehensive R Archive Network (CRAN) Task View provides a substantive list of R packages that delivers Bayesian inference for point-referenced spatial data, among which, **geoR** (Ribeiro Jr, Diggle, Christensen, Schlather, Bivand, and Ripley 2024), **geoRglm** (Christensen and Ribeiro 2002), **spBayes** (Finley, Banerjee, and Carlin 2007), **ramps** (Smith, Yan, and Cowles 2008), **spTimer** (Bakar and Sahu 2015), **INLA** (Rue, Martino, and Chopin 2009) and **FRK** (Zammit-Mangion and Cressie 2021; Sainsbury-Dale, Zammit-Mangion, and Cressie 2024) are worth mentioning. These packages either fits a fully Bayesian model using MCMC, or builds upon methodologies based on discretized spatial domains and have limited support for non-Gaussian data, especially since **geoRglm** is no longer being actively developed. We introduce the R package **spStack** (Pan and Banerjee 2024) that delivers fast Bayesian inference, completely avoiding MCMC algorithms, for Gaussian as well as non-Gaussian point-referenced spatial data, that includes Poisson, binomial and binary data, using stacking of predictive densities. **spStack**, to the best of our knowledge, is the first to implement Bayesian predictive stacking for analysis of spatial data. The package is primarily written in C++ for speed, accompanied by calls to Fortran routines offered by efficient linear algebra libraries for optimized matrix computations. With the growing capacity of parallel computing, development of **spStack** enhances practicality of Bayesian inference for geostatistical models over sequential algorithms like MCMC.

2. Bayesian spatial regression models

Consider a spatial process as an uncountable set of random variables, say $\{z(s) : s \in \mathcal{S}\}$, over a spatial region of interest $\mathcal{S} \in \mathbb{R}^d$, which is endowed with a probability law specifying the joint distribution for any finite sample from that set. Let $y(s)$ denote the outcome at $s \in \mathcal{S}$ endowed with a probability law from the natural exponential family, which we denote by

$$y(s) \sim \text{EF}(x(s)^\top \beta + z(s); b, \psi_y) \tag{1}$$

for some positive parameter $b > 0$ and unit log partition function ψ_y (Section A.1). Various forms of the unit log partition function $\psi(\cdot)$ and the scalar b denote different distributions from the natural exponential family. For example, $\psi_1(t) = t^2$ with $b = 1/2\sigma^2$ correspond to Gaussian outcomes with variance σ^2, $\psi_2(t) = e^t$ with $b = 1$ correspond to Poisson counts, and $\psi_3(t) = \log(1 + e^t)$ with $b = 1$ or $b = m(s) > 1$ correspond to binary outcomes or binomial counts with $m(s)$ number of trials. Fixed effects regression and spatial dependence is introduced in the natural parameter, $\eta(s) = x(s)^\top \beta + z(s)$ where $x(s)$ is a $p \times 1$ vector of predictors referenced with respect to s, β is a $p \times 1$ vector of slopes measuring the trend, $z(s) \sim \text{GP}(0, \sigma_z^2 R(\cdot, \cdot; \theta_{\text{sp}}))$ is a zero-centered spatial Gaussian process on $\mathcal{S}$ specified by spatial variance parameter σ_z^2 and a spatial correlation function $R(\cdot, \cdot; \theta_{\text{sp}})$ with θ_{sp} consisting of spatial decay (ϕ) and smoothness (ν) parameters. Currently, **spStack** supports only the isotropic Matérn correlation function,

$$R(s, s'; \theta_{\text{sp}}) = \frac{(\phi|s - s'|)^\nu}{2^{\nu-1}\Gamma(\nu)} K_\nu\left(\phi|s - s'|\right) \ , \tag{2}$$

where $|s - s'|$ is the Euclidean distance between s and s'. The function $\Gamma(\cdot)$ denotes the gamma function, and K_ν is the modified Bessel function of the second kind of order ν which may be fractional (Abramowitz and Stegun 1965, Chapter 10). We define a data $\mathcal{D}$ by the triplet $\mathcal{D} = \{y, X, \chi\}$ which collects the quantities that are known to a practitioner.

2.1. Conjugate Bayesian Gaussian spatial model

Let $\chi = \{s_1, \ldots, s_n\} \in \mathcal{S}$ be a set of n distinct spatial locations yielding measurements $y = (y(s_1), \ldots, y(s_n))^\top$ which are modeled as

$$y(s_i) = x(s_i)^\top \beta + z(s_i) + \epsilon(s_i), \quad \epsilon(s_i) \overset{\text{iid}}{\sim} N(0, \sigma^2), \tag{3}$$

with known values of predictors at these locations collected in a $n \times p$ full column rank matrix $X = [x(s_1), \ldots, x(s_n)]^\top$ and $\epsilon(s_i)$ for each i denotes independent and identically distributed measurement errors. Let $z = (z(s_1), \ldots, z(s_n))^\top$ denote the realization of the spatial process over χ and $R(\chi; \theta_{\text{sp}}) = (R(s_i, s_j; \theta_{\text{sp}}))_{1 \le i,j \le n}$ be the $n \times n$ spatial correlation matrix constructed from (2). For ease of notation, we drop χ and θ_{sp} from $R(\chi; \theta_{\text{sp}})$ and simply write R for the spatial correlation matrix. Zhang, Tang, and Banerjee (2024) construct a conjugate Bayesian hierarchical model

$$\begin{aligned} y \mid \beta, z, \sigma_z^2 &\sim \mathcal{N}(X\beta + z, \delta^2 \sigma_z^2 I_n), \quad z \mid \sigma_z^2 \sim \mathcal{N}(0, \sigma_z^2 R(\chi; \theta_{\text{sp}})), \\ \beta \mid \sigma_z^2 &\sim \mathcal{N}(\mu_\beta, \sigma_z^2 V_\beta), \quad \sigma_z^2 \sim \mathcal{IG}(a_\sigma, b_\sigma) \ , \end{aligned} \tag{4}$$

where we consider a fixed value of the noise-to-spatial variance ratio $\delta^2 = \sigma^2/\sigma_z^2$, the process parameters in θ_{sp}, and the hyperparameters μ_β, V_β, a_σ, b_σ. We define a model M within the family of hierarchical models (4) by specifying fixed values of θ_{sp}, δ^2 and the hyperparameters. From (4), we derive the posterior distribution of (γ, σ_z^2), where $\gamma = (\beta^\top, z^\top)^\top$ as

$$\begin{aligned} p(\gamma, \sigma_z^2 \mid \mathcal{D}, M) &= p(\sigma_z^2 \mid \mathcal{D}, M) \times p(\gamma \mid \sigma_z^2, \mathcal{D}, M) \\ &= \mathcal{IG}(\sigma_z^2; a_*, b_*) \times \mathcal{N}(\gamma; \hat{\gamma}, \sigma_z^2 K_*), \end{aligned} \tag{5}$$

where $\hat{\gamma} = K_* X_*^\top V_*^{-1} y_*$, $y_* = (y^\top, \mu_\beta^\top, 0_n^\top)^\top$, $a_* = a_\sigma + n/2$, $b_* = b_\sigma + \frac{1}{2}(y_* - X_*\hat{\gamma})^\top V_*^{-1}(y_* - X_*\hat{\gamma})$, $K_*^{-1} = X_*^\top V_*^{-1} X_*$, $X_* = [(X : I_n); I_{p+n}]$, and $(2n+p) \times (2n+p)$ block-diagonal matrix $V_* = \text{blkdiag}(\delta^2 I_n, V_\beta, R)$. This representation of the posterior distribution follows from the augmented linear system framework (see Banerjee 2020, Section 3.1). However, matrix computation under this representation is suboptimal as it requires storage and handling of unnecessarily large matrices and hence, we pursue a computationally efficient alternative.

Sampling from the posterior distribution

The posterior distribution (5) is further decomposed into

$$\begin{aligned} p(\beta, z, \sigma_z^2 \mid \mathcal{D}, M) &= p(\sigma_z^2 \mid \mathcal{D}, M) \times p(\beta \mid \sigma_z^2, \mathcal{D}, M) \times p(z \mid \beta, \sigma_z^2, y, M) \\ &= \mathcal{IG}(\sigma_z^2; a_*, b_*) \times \mathcal{N}(\beta; Bb, \sigma_z^2 B) \times \mathcal{N}(z; Cc, \sigma_z^2 C), \end{aligned} \tag{6}$$

where $a_* = a_\sigma + n/2$, $b_* = b_\sigma + (y^\top V_y^{-1} y + \mu_\beta^\top V_\beta^{-1} \mu_\beta - b^\top B b)/2$, $B^{-1} = X^\top V_y^{-1} X + V_\beta^{-1}$, $b = X^\top V_y^{-1} y + V_\beta^{-1} \mu_\beta$, $C^{-1} = R^{-1} + (1/\delta^2) I_n$, $c = (y - X\beta)/\delta^2$ and $V_y = R + \delta^2 I_n$. Special consideration is placed on computing C since the Cholesky decomposition $\texttt{chol}(R)$ is deemed numerically unstable under different correlation kernels, especially in presence of sampled locations in close proximity. We avoid this by utilizing the identity

$$C = \left(R^{-1} + (1/\delta^2) I_n\right)^{-1} = \delta^2 (R + \delta^2 I_n)^{-1} R = \delta^2 V_y^{-1} R.$$

This representation is particularly useful since it helps express all parameters involved in the posterior in terms of V_y. Equation 6 facilitates an efficient *composition sampling* strategy to obtain exact posterior samples. To elucidate further, first we sample $\sigma_z^{2(\ell)}$ for $\ell = 1, \ldots, N_s$ from $p(\sigma_z^2 \mid \mathcal{D})$. For each ℓ, we sample $\beta^{(\ell)}$ from $p(\beta \mid \sigma_z^{2(\ell)}, \mathcal{D})$, and, finally, draw $z^{(\ell)}$ from $p(z \mid \beta^{(\ell)}, \sigma_z^{2(\ell)}, \mathcal{D})$. The last step requires computing a triangular solve of a $n \times n$ matrix to obtain C and, subsequently find its Cholesky factor. Hence, the computation required to sample from (5) is dominated by two Cholesky decompositions $L_y \leftarrow \texttt{chol}(V_y)$ and $\texttt{chol}(C)$, and, two triangular solves $\texttt{trsolve}(L_y^\top, \texttt{trsolve}(L_y, R))$, where $\texttt{trsolve}(A, B)$ refers to solving the triangular system $AX = B$ for matrices X and B, and triangular matrix A. Thus, it requires $O(2n^3/3 + n^3)$ i.e., $O(5n^3/3)$ floating point operations (flops) and $O(2n^2)$ storage. This is implemented in the `spLMexact()` function offered by **spStack**.

We demonstrate `spLMexact()` on the synthetic dataset `simGaussian` available in **spStack**.

```
R> library("spStack")
R> data(simGaussian)
```

The dataset is simulated from the model (3) using the `sim_spData()` function with `family = "gaussian"`, $n = 500$ locations sampled uniformly on an unit square, number of predictors including intercept $p = 2$, fixed effects $\beta = (2, 5)^\top$, spatial effects drawn from a Gaussian process under the Matérn correlation kernel with $\phi = 2$, $\nu = 0.5$, $\sigma_z^2 = \sigma^2 = 0.4$ which implies $\delta^2 = 1$. Code to reproduce all synthetic data is available in the package manual. If not using the default priors, it can be set up using the following code.

```
R> muBeta <- c(0, 0)
R> VBeta <- diag(1E3, 2)
R> sigmaSqIGa <- 2
R> sigmaSqIGb <- 2
R> prior_list <- list(beta.norm = list(muBeta, VBeta),
+                     sigma.sq.ig = c(sigmaSqIGa, sigmaSqIGb))
```

The code below runs the analysis under the fixed values $\phi = 3$, $\nu = 0.75$ and $\delta^2 = 0.8$.

```
R> set.seed(1729)
R> mod1 <- spLMexact(y ~ x1, data = simGaussian,
R>                   coords = as.matrix(simGaussian[, c("s1", "s2")]),
R>                   cor.fn = "matern",
R>                   priors = prior_list,
R>                   spParams = list(phi = 3, nu = 0.75),
R>                   noise_sp_ratio = 0.8,
R>                   n.samples = 1000, verbose = FALSE)
```

```
----------------------------------------
        Model description
----------------------------------------
Model fit with 500 observations.

Number of covariates 2 (including intercept).

Using the matern spatial correlation function.

Priors:
        beta: Gaussian
        mu: 0.00  0.00
        cov:
        1000.00  0.00
        0.00     1000.00

        sigma.sq: Inverse-Gamma
        shape = 2.00, scale = 2.00.

Spatial process parameters:
        phi = 3.00, and, nu = 0.75.
Noise-to-spatial variance ratio = 0.80.

Number of posterior samples = 1000.
----------------------------------------
```

The following code summarizes the 95% credible intervals of the fixed effects.

```
R> beta.post <- mod1$samples$beta
R> rownames(beta.post) <- mod1[["X.names"]]
R> print(t(apply(beta.post, 1,
+                  function(x) quantile(x, c(0.025, 0.5, 0.975)))))
```

```
                2.5%      50%    97.5%
(Intercept) 1.343554 2.072259 2.753898
x1          4.949188 5.002247 5.058618
```

We observe that the posterior of the intercept and the slope for the variable `x1` has concentrated around its true value 2 and 5, respectively. Running `help("spLMexact")` reveals its documentation. The Examples section of the documentation provides additional code to visualize and compare the interpolated spatial surfaces of the true spatial random effects with which the data was simulated and the posterior median of the spatial effects.

Spatial prediction

Let $\tilde{\chi} = \{\tilde{s}_1, \dots, \tilde{s}_{\tilde{n}}\} \in \mathcal{S} \setminus \chi$ be a collection of $\tilde{n}$ new locations in $\mathcal{S}$, where we wish to predict the latent spatial process as well as the response. Let $\tilde{z}$ and $\tilde{y}$ be the $\tilde{n} \times 1$ vectors with ith elements $z(\tilde{s}_i)$ and $\tilde{y}(s_i)$ denoting the spatial effect and the response at location $\tilde{s}_i$. Let $\tilde{n} \times p$ full column rank matrix $\tilde{X} = [x(\tilde{s}_1), \dots, x(\tilde{s}_{\tilde{n}})]^\top$ collects the known predictors at the new locations $\tilde{\chi}$, and $\tilde{n} \times \tilde{n}$ matrix $\tilde{R} = R(\tilde{\chi}; \theta_{\text{sp}})$ is the correlation matrix corresponding to $\tilde{\chi}$ and $n \times \tilde{n}$ matrix $J = R(\chi, \tilde{\chi}; \theta_{\text{sp}})$ denote the cross-covariance matrix between the sets of locations χ and $\tilde{\chi}$. Spatial predictive inference follows from the posterior distribution

$$p(\tilde{y}, \tilde{z} \mid \mathcal{D}, M) = \int p(\tilde{y} \mid \tilde{z}, \beta, \sigma_z^2, M)\, p(\tilde{z} \mid z, \sigma_z^2, M)\, p(\beta, z, \sigma_z^2 \mid \mathcal{D}, M)\, d\beta\, dz\, d\sigma_z^2, \tag{7}$$

which is a multivariate t distribution with location $\tilde{\mu}$ and scale matrix $(b_*/a_*)\tilde{V}$, where $2\tilde{n} \times 1$ vector $\tilde{\mu} = W\hat{\gamma}$, $2\tilde{n} \times 2\tilde{n}$ matrix $\tilde{V} = WK_*W^\top + \tilde{K}$ with $2\tilde{n} \times (n+p)$ matrix $W = [(\tilde{X} : J^\top R^{-1}); (0 : J^\top R^{-1})]$, $2\tilde{n} \times 2\tilde{n}$ matrix $\tilde{K}^{-1} = [(R_{\tilde{z}|z}^{-1} + \frac{1}{\delta^2} I_{\tilde{n}} : -\frac{1}{\delta^2} I_{\tilde{n}}); (-\frac{1}{\delta^2} I_{\tilde{n}} : \frac{1}{\delta^2} I_{\tilde{n}})]$, and $\tilde{n} \times \tilde{n}$ matrix $R_{\tilde{z}|z} = \tilde{R} - J^\top R^{-1} J$ (see, Zhang *et al.* 2024). These closed form expressions of the posterior predictive densities are particularly useful when predictive inference is sought without fitting the model. However, if posterior samples $\{\beta^{(\ell)}, z^{(\ell)}, \sigma_z^{2(\ell)}; \ell = 1, \dots, N_s\}$ are available, predictive inference for $\tilde{z}$ and $\tilde{y}$ may proceed by drawing $\tilde{z}^{(\ell)}$ from $p(\tilde{z} \mid z^{(\ell)}, \sigma_z^{2(\ell)}, M)$ which is essentially $\mathcal{N}(J^\top R^{-1} z^{(\ell)}, \sigma_z^{2(\ell)} R_{\tilde{z}|z})$, and subsequently, drawing $\tilde{y}^{(\ell)}$ from $p(\tilde{y} \mid \tilde{z}^{(\ell)}, \beta^{(\ell)}, \sigma_z^{2(\ell)}, M)$ that is $\mathcal{N}(\tilde{X}\beta^{(\ell)} + \tilde{z}^{(\ell)}, \delta^2 \sigma_z^{2(\ell)} I_{\tilde{n}})$ respectively, for each ℓ. Prediction of the latent spatial process requires additional one Cholesky decomposition `chol`(R), and two triangular solves `trsolve`(`chol`(R), $z^{(\ell)}$) and `trsolve`(`chol`(R), J).

2.2. Conjugate Bayesian non-Gaussian spatial model

Given a fixed set of n locations $\chi = \{s_1, \dots, s_n\}$ in $\mathcal{S}$, let $y = (y(s_1), \dots, y(s_n))^\top \in \mathscr{Y}^n$ is the vector of observed outcomes, where $\mathscr{Y}$ denotes the support of the outcome variable. Following

Bradley and Clinch (2024), we introduce a conjugate Bayesian hierarchical spatial model as

$$\begin{aligned}y(s_i) \mid \beta, z(s_i), \xi_i, \mu_i &\overset{\text{ind.}}{\sim} \text{EF}\left(x(s_i)^\top\beta + z(s_i) + \xi_i - \mu_i; b(s_i), \psi_y\right), \quad i = 1, \ldots, n\\ z \mid \sigma_z^2 &\sim \mathcal{N}(0, \sigma_z^2 R(\chi; \theta_{\text{sp}})), \quad \beta \mid \sigma_\beta^2 \sim \mathcal{N}(0, \sigma_\beta^2 V_\beta),\\ \xi \mid \beta, z, \mu, \sigma_\xi &\sim \text{GCM}_\text{c}(\tilde{\mu}_\xi, H_\xi, \alpha_\xi, \kappa_\xi; \psi_\xi),\\ \sigma_\beta^2 &\sim \mathcal{IG}(\nu_\beta/2, \nu_\beta/2), \quad \sigma_z^2 \sim \mathcal{IG}(\nu_z/2, \nu_z/2)\,,\end{aligned} \tag{8}$$

where $x(s_i)$ is a $p \times 1$ vector of predictors, β is the corresponding $p \times 1$ vector of fixed effects, $z = (z(s_1), \ldots, z(s_n))^\top$ is $n \times 1$ with each $z(s_i)$ being a realization of the spatial process with under correlation kernel (2) at location s_i, ψ_y is either ψ_2 or ψ_3, denoting Poisson or binary/binomial data, and $b = (b(s_1), \ldots, b(s_n))^\top$ is known. The conditional prior for ξ is a GCM_c distribution (see, Section A.2) with the $2n \times n$ matrix $H_\xi = [I_n; \sigma_\xi I_n]$, location parameter $\tilde{\mu}_\xi = H_\xi((\mu - X\beta - z)^\top, \mu_\xi^\top)^\top$, where $\mu = (\mu_1, \ldots, \mu_n)^\top$ is $n \times 1$, shape and scale parameters $\alpha_\xi = (\alpha_\epsilon 1_n^\top, 0_n^\top)^\top$ and $\kappa_\xi = (\kappa_\epsilon 1_n^\top, (1/2)1_n^\top)^\top$, respectively, for some fixed positive reals α_ϵ and κ_ϵ. The unit log partition function ψ_ξ is defined as $\psi_\xi(h) = (\psi_y(h_1)^\top, \psi_1(h_2)^\top)^\top$ for any $h = (h_1, h_2)$ with $h_1, h_2 \in \mathbb{R}^n$, where $\psi_y(\cdot)$ and $\psi_1(\cdot)$ operate element-wise on their respective arguments. Gather the fixed and random effects into the $(2n + p) \times 1$ parameter $\gamma = (\xi^\top, \beta^\top, z^\top)^\top$. Let $\tilde{\mu} = (\mu^\top, \mu_\gamma^\top)^\top$ be the $(3n+p) \times 1$ vector obtained by combining μ with the location parameters $\mu_\gamma = (\mu_\xi^\top, 0_{n+p}^\top)^\top$, where 0_m denotes a $m \times 1$ zero vector. Let $q = -Q^\top\tilde{\mu}$, where Q is $(3n+p) \times n$ and obtained from the decomposition $QQ^\top = I_{3n+p} - P_H$ with the columns of Q being the n unit norm orthogonal eigenvectors of $I_{3n+p} - P_H$ corresponding to eigenvalue 1, where $P_H = H(H^\top H)H^\top$. Hence, $Q^\top Q = I_n$ and $H^\top Q = 0$. Furthermore, $H = [(I_n : X : I_n); L^{-1}]$ is $(3n + p) \times (2n + p)$, X is $n \times p$ with $x(s_i)^\top$ as its i-th row, and $(2n+p) \times (2n+p)$ matrix $L = \text{blkdiag}(\sigma_\xi I_n, L_\beta, L_z)$, where $L_\beta = \texttt{chol}(V_\beta)$ and $L_z = \texttt{chol}(R)$ are lower-triangular Cholesky factors of V_β and $R = R(\chi; \theta_{\text{sp}})$. We define a model M within the family of hierarchical models (8) by specifying fixed values of θ_{sp}, the boundary adjustment parameters α_ϵ and κ_ϵ, and the hyperparameters V_β, ν_β, ν_z and σ_ξ.

We work with an improper prior on q given by $p(q) \propto 1$ which follows from assuming a vague prior on μ (see, Proposition A1 and Lemma A1 in Pan, Zhang, Bradley, and Banerjee 2024, for technical details). These specifications yield the posterior distribution

$$p\left((\gamma^\top, q^\top)^\top \mid \mathcal{D}, M\right) \propto \text{GCM}\left((\gamma^\top, q^\top)^\top; 0_{3n+p}, V_*, \alpha_*, \kappa_*; \psi_*\right)\,, \tag{9}$$

where $V_*^{-1} = [H : Q]$, and the shape and scale parameters are $\alpha_* = ((y + \alpha_\epsilon 1_n)^\top, 0_{2n+p}^\top)^\top$ and $\kappa_* = ((b + \kappa_\epsilon 1_n)^\top, (1/2)1_n^\top, (1/2)(\nu_\beta + 1)1_p^\top, (1/2)(\nu_z + 1)1_n^\top)^\top$. The unit log partition function ψ_* in (9) is $\psi_*(h) = (\psi_y(h_1)^\top, \psi_1(h_2)^\top, \psi_4(h_3; \nu_\beta)^\top, \psi_4(h_4; \nu_z)^\top)^\top$ for some $h = (h_1^\top, h_2^\top, h_3^\top, h_4^\top)^\top$ with $h_1, h_2, h_4 \in \mathbb{R}^n$, $h_3 \in \mathbb{R}^p$, where the log partition functions operate element-wise on the arguments (see Theorem A1 in Pan *et al.* 2024). The function $\psi_4(\cdot; \cdot)$ corresponds to the log partition function of the Student's t distribution family (see Section A.1) and appears in (9) as a result of integrating out σ_β^2 and σ_z^2 to obtain marginal priors $p(\beta)$ and $p(z)$. Generalized linear models typically assume that q is zero, which yields the posterior distribution $p(\gamma \mid \mathcal{D}, M) \propto \text{GCM}_\text{c}(\gamma; \tilde{\mu}, H, \alpha_*, \kappa_*; \psi_*)$, from which we cannot sample directly (see Section A.2) except for some special cases of ψ_y (e.g., Gaussian). Finally, we remark that μ is crucial in producing the posterior distribution within the GCM family and, hence, unlike in traditional generalized linear models, cannot be excluded from (8).

Sampling from the posterior distribution

To obtain samples of $\gamma = (\xi^\top, \beta^\top, z^\top)^\top$ from their posterior distribution (9), we first draw $v^{(\ell)} \sim \text{GCM}(0_{3n+p}, I_{3n+p}, \alpha_*, \kappa_*; \psi_*)$ which is equivalent to sampling independent DY random variables, for each ℓ. To elaborate further, $v^{(\ell)} = (v_\eta^{(\ell)\top}, v_\xi^{(\ell)\top}, v_\beta^{(\ell)\top}, v_z^{(\ell)\top})^\top$ is made up of $n \times 1$ vector $v_\eta^{(\ell)} = (v_{\eta,1}^{(\ell)}, \ldots, v_{\eta,1}^{(\ell)})^\top$ which is a draw from the posterior distribution of the $n \times 1$ natural parameter $\eta = X\beta + z + \xi - \mu$, with i-th element $v_{\eta,i}^{(\ell)} \sim \text{DY}(y_i + \alpha_\epsilon, b(s_i) + \kappa_i; \psi_y)$ for each i. For example, in case of Poisson data, the DY random variable $v_{\eta,i}^{(\ell)}$ corresponds to independent log-gamma draws for each i (see, Bradley, Holan, and Wikle 2020). Moreover, $v_\xi^{(\ell)}$ is $n \times 1$ and made up of independent draws from $\mathcal{N}(0, \sigma_\xi^2)$, $p \times 1$ vector $v_\beta^{(\ell)}$ and $n \times 1$ vector $v_z^{(\ell)}$ are made up of independent draws from Student's t distribution with degrees of freedom ν_β and ν_z respectively. Subsequently, since $[H : Q]^{-1} = [H(H^\top H)^{-1} : Q]^\top$, we compute

$$\gamma^{(\ell)} \leftarrow (H^\top H)^{-1} H^\top v^{(\ell)}, \quad q^{(\ell)} \leftarrow Q^\top v^{(\ell)} , \tag{10}$$

to obtain samples $(\gamma^{(\ell)\top}, q^{(\ell)\top})^\top$ from (9). The projection expression in (10) for obtaining replicates $\gamma^{(\ell)}$ merits special attention due to its similarity with the posterior distribution of the fixed effects and the latent process in (5) under the augmented linear model framework. We develop an efficient algorithm for evaluation of $(H^\top H)^{-1} H^\top v^{(\ell)}$ for each ℓ, which proceeds by a *priming* step and, subsequently, a *projection* step -

- In the *priming* step, we exploit the structured nature of the matrix H and use block-matrix inversion strategies to compute certain matrices that are related to different components of $(H^\top H)^{-1}$, thus avoiding explicit inversion.
- In the *projection* step, we repetitively use the matrices obtained from the *priming* step to evaluate (10) and obtain $\gamma^{(\ell)}$ for each $v^{(\ell)}$ using a `for` loop over ℓ.

Note that, the *priming* step is expensive as it requires $O(n^3)$ flops, and, hence, we design the sampling algorithm in a way such that the *priming* step needs to be evaluated only once and its output can be utilized for subsequent posterior sampling. This is implemented in the `spGLMexact()` function offered by **spStack**. We would also like to emphasize that `spGLMexact()` is the first to implement exact sampling strategy for spatial non-Gaussian data using the generalized conjugate multivariate framework (Bradley and Clinch 2024).

We demonstrate `spGLMexact()` on the synthetic spatial count dataset `simPoisson` available in **spStack**. `simPoisson` is simulated from the model (1) using the `sim_spData()` function with `family = "poisson"`, $n = 500$ locations sampled uniformly on an unit square, $p = 2$, fixed effects $\beta = (2, -0.5)^\top$, spatial effects drawn from a Gaussian process under the Matérn correlation kernel with $\phi = 5$, $\nu = 0.5$, and spatial variance $\sigma_z^2 = 0.4$. The following code runs the analysis by drawing 1000 samples from (9) conditional on $\phi = 4$, $\nu = 0.4$ and $\alpha_\epsilon = 0.5$.

```
R> data(simPoisson)
R> mod2 <- spGLMexact(y ~ x1, data = simPoisson, family = "poisson",
+                     coords = as.matrix(simPoisson[, c("s1", "s2")]),
+                     cor.fn = "matern", spParams = list(phi = 4, nu = 0.4),
+                     boundary = 0.5, n.samples = 1000, verbose = TRUE)
```

```
----------------------------------------
        Model description
----------------------------------------
Model fit with 500 observations.

Family = poisson.

Number of covariates 2 (including intercept).

Using the matern spatial correlation function.

Priors:
        beta: Gaussian
        mu: 0.00  0.00
        cov:
        100.00  0.00
        0.00    100.00

        sigmaSq.beta ~ IG(nu.beta/2, nu.beta/2)
        sigmaSq.z ~ IG(nu.z/2, nu.z/2)
        nu.beta = 2.10, nu.z = 2.10.
        sigmaSq.xi = 0.10.
        Boundary adjustment parameter = 0.50.

Spatial process parameters:
        phi = 4.00, and, nu = 0.40.

Number of posterior samples = 1000.
----------------------------------------
```

The following code summarizes the 95% credible intervals of the fixed effects. We note that the posterior medians of the intercept and the slope of the variable `x1` is sitting around their corresponding true values.

```
R> beta.post <- mod2$samples$beta
R> rownames(beta.post) <- mod1[["X.names"]]
R> print(t(apply(beta.post, 1,
+                function(x) quantile(x, c(0.025, 0.5, 0.975)))))

                  2.5%        50%      97.5%
(Intercept)  0.5966932  2.0912370  3.6759803
x1          -0.5872058 -0.5079498 -0.4137453
```

We note that GCM customarily uses posterior samples of β and z to draw samples from the posterior predictive distribution $p(x(s)^\top\beta + z(s) + \xi - \mu \mid \mathcal{D}, M)$ for any $s \in \chi$ by simply considering $x(s)^\top\beta^{(\ell)} + z^{(\ell)}(s)$, where $\{\beta^{(\ell)}, z^{(\ell)}(s)\}$ are replicates from (9), which implicitly estimates ξ and μ to be zero after integrating them out from (9) (Bradley and Clinch 2024).

Spatial prediction

Given data observed at χ, let $\tilde{\chi} = \{\tilde{s}_1, \dots, \tilde{s}_{\tilde{n}}\} \subset \mathcal{S} \setminus \chi$ be a collection of $\tilde{n}$ new locations in $\mathcal{S}$, where we wish to predict the response and the latent spatial processes. Let $\tilde{y}$ and $\tilde{z}$ be the $\tilde{n} \times 1$ vectors with ith elements $y(\tilde{s}_i)$ and $z(\tilde{s}_i)$ respectively. For a given model M, which entails a fixed value of θ_{sp} and some auxiliary model parameters, spatial predictive inference evaluates the posterior predictive distribution,

$$p(\tilde{y}, \tilde{z} \mid \mathcal{D}, M) = \int p(\tilde{y} \mid \beta, \tilde{z}, \mathcal{D})\, p(\tilde{z} \mid z, M)\, p(\beta, z \mid \mathcal{D}, M) d\beta dz \; . \tag{11}$$

Unlike the Gaussian case, (11) is not analytically tractable and hence, sampling from (11) is facilitated by first drawing $\{\beta^{(\ell)}, z^{(\ell)}\}$ from $p(\beta, z \mid \mathcal{D}, M)$ as described in Section 2.2.1 and then, for each drawn value $z^{(\ell)}$, $\tilde{z}^{(\ell)}$ is drawn from $p(\tilde{z} \mid z, M)$. Further, for each posterior sample $\beta^{(\ell)}$ and $\tilde{z}^{(\ell)}$, we draw $\tilde{y}^{(\ell)}$ from $p(\tilde{y} \mid \beta, \tilde{z}, \mathcal{D})$ with μ and ξ in (8) set to 0. This yields samples $\{\tilde{y}^{(\ell)}, \tilde{z}^{(\ell)}\}$ from (11). Under (8), the marginal distribution of the $(n + \tilde{n}) \times 1$ vector $(z^\top, \tilde{z}^\top)^\top$ is a multivariate t-distribution $t_{\nu_z}(0_n, \tilde{V}_z)$ with ν_z degrees of freedom, location parameter $0_{n+\tilde{n}}$ and $(n + \tilde{n}) \times (n + \tilde{n})$ scale matrix $\tilde{V}_z = [(R : J); (J^\top : \tilde{R})]$, with symbols as defined previously in Section 2.1.2. This yields

$$\tilde{z} \mid z, M \sim t_{n+\nu_z}\left(J^\top R^{-1} z, \; \frac{z^\top R^{-1} z + \nu_z}{n + \nu_z} (\tilde{R} - J^\top R^{-1} J)\right) \; . \tag{12}$$

It is worth noticing that, the scale matrix contains the factor $(z^\top R^{-1} z + \nu_z)/(n + \nu_z)$ which is directly related to the Mahalanobis distance of z implying that the dispersion is enlarged in presence of extreme values of z. The degrees of freedom also increases by a factor n which means that, the more data we have, the less heavy-tailed $p(\tilde{z} \mid z, M)$ becomes (Ding 2016).

3. Predictive stacking

Following the generalized Bayesian stacking framework in Yao *et al.* (2018), we devise a stacking algorithm for geostatistical analysis, which we refer to as *predictive stacking*. Let $\mathcal{M} = \{M_1, \dots, M_G\}$ be a collection of G candidate models, where each M_g corresponds to a model with fixed values of certain parameters, as defined in Sections 2.1 and 2.2 for Gaussian and non-Gaussian settings, respectively. Traditional Bayesian model averaging (BMA) asymptotically chooses a single model in $\mathcal{M}$ that is closest in terms of Kullback-Leibler (KL) divergence to the true data generating model M_0, and hence, is flawed under the $\mathcal{M}$-open setting where $\mathcal{M}$ does not contain M_0 (Bernardo and Smith 1994). Yao *et al.* (2018) effectively mitigates this issue by generalizing stacking to combine Bayesian predictive distributions. Given the complex nature of spatial dependence in (1), the assumption that $\mathcal{M}$ contains the true model M_0 is rarely tenable in practical geostatistics and we prefer stacking to BMA. Using predictive stacking for Bayesian inference of Gaussian and non-Gaussian data are implemented in the functions `spLMstack()` and `spGLMstack()` respectively. Detailed example code for implementing these core functions are discussed in Section 5.

3.1. Stacking algorithm

The stacking algorithm finds a $\tilde{p}$ in $\mathcal{C} = \{\sum_{g=1}^G w_g\, p(\cdot \mid \mathcal{D}, M_g) : \sum_{g=1}^G w_g = 1, w_g \geq 0, \forall g\}$, such that, its KL-divergence to $p_t(\cdot, \mathcal{D})$, the true data generating model, is minimized. We define the stacking weights w as the solution to the optimization problem

$$\max_{w_1,\dots,w_G} \frac{1}{n}\sum_{i=1}^{n} \log \sum_{g=1}^{G} w_g p(y(s_i) \mid \mathcal{D}_{-i}, M_g) \quad \text{s.t.} \quad w_g \geq 0, \sum_{g=1}^{G} w_g = 1 \;, \tag{13}$$

where $\mathcal{D}_{-i} = \{y_{-i}, X_{-i}, \chi_{-i}\}$ denotes the data excluding the i-th observation, for $i = 1, \dots, n$. The $(n-1) \times 1$ vector y_{-i} collects the outcomes except the i-th, $(n-1) \times p$ matrix X_{-i} denote the matrix of predictors X with the i-th row deleted, and $\chi_{-i} = \chi \setminus \{s_i\}$. This follows from the result that minimizing $\mathrm{KL}\left(\tilde{p}(\cdot \mid \mathcal{D}), p_t(\cdot \mid \mathcal{D})\right)$ under the constraint $\tilde{p} \in \mathcal{C}$ is asymptotically equivalent to the optimization problem in (13) (see, Le and Clarke 2017; Clyde and Iversen 2013). The optimization task in (13) falls within the class of convex programming and can be implemented using suitable modeling tools and solvers. While the `stacking_weights()` function from the **loo** (Vehtari, Gabry, Magnusson, Yao, Bürkner, Paananen, and Gelman 2024a) package offers this utility, the returned weights have often been found to be suboptimal, thus impacting its reliability. To address these limitations, we propose the function `get_stacking_weights()`, which has the arguments

- `log_loopd`: an $n \times G$ matrix of leave-one-out predictive densities in log scale, where n is the sample size and, G is the number of candidate models.
- `solver`: a quoted keyword specifying the solver to use, e.g., `"ECOS"`.

While `stacking_weights()` from **loo** uses general purpose optimization tools provided by `optim()` of the **stats** package, `get_stacking_weights()` in **spStack** uses the **CVXR** package, which brings the functionalities of the popular convex optimization software **CVX** in R. It proceeds by first verifying the convexity of the problem using disciplined convex programming (DCP) before passing it to a solver (Fu, Narasimhan, and Boyd 2020). The `solver` argument of `get_stacking_weights()` allows the user to interface with different solvers of their choice, even beyond the available solvers in **CVXR**. For example, the user can use the following example code to use the commercial solver **Mosek** for finding optimal stacking weights. The code assumes that a $n \times G$ matrix containing log leave-one-out predictive densities for all G candidate models is stored in the variable `loopd_mat`.

```
library("Rmosek")
w_hat <- get_stacking_weights(loopd_mat, solver = "MOSEK")
```

The returned object `w_hat` is a named list with `weights` containing a $G \times 1$ numeric vector of the optimal stacking weights, and `status` containing a character specifying the solver status, e.g., `"optimal"` implies a successful search. It must be noted that, using **Mosek** software requires a license and installation of the package **Rmosek** (ApS 2024) to interface from R.

3.2. Leave-one-out predictive densities

It is clear that, a critical step in solving for the stacking weights is the evaluation of the leave-one-out predictive densities $p(y(s_i) \mid \mathcal{D}_{-i}, M_g)$ for each $i = 1, \dots, n$ and $g = 1, \dots, G$. This step is non-trivial from a computational viewpoint, since, a naive approach would require either repeated Cholesky decompositions or refitting the model n times for each g. Note that, the computation required to sample from the respective posterior distributions under the Gaussian and the non-Gaussian settings, is dominated by Cholesky factorizations of $n \times n$ matrices related to the spatial correlation matrix R, which requires $O(n^3/3)$ flops. The naive approach for finding leave-one-out predictive densities (LOO-PD) entails $O(n^4)$ flops which is an impractical computational cost even for moderately large datasets. Hence, we devise computationally efficient strategies to find the LOO-PD for all G candidate models.

We pursue strategies like row-deletion updates of a Cholesky factor (Tae Yoon Kim and Cox 2002), which computes the factor `chol`(A_{-I}) from `chol`(A), where A is a $n \times n$ positive-definite matrix, I is a subset of $\{1, \ldots, n\}$ of size $m \geq 1$ comprising of m consecutive indices, and, A_{-I} denotes the $(n-m) \times (n-m)$ submatrix of A put together by excluding the row and columns with indices in I. Such updates can be reformulated as m rank-one updates of a submatrix of the factor `chol`(A) and can be done in $O(m(n - I_{\max})^2)$, where $I_{\max} = \max(I)$. For ease of notation, we use the functions `cholDelUpdate()` and `cholBlockDelUpdate()` to denote these utilities for the cases $m = 1$ and $1 < m < n-1$, respectively. More details on these algorithms are in Section 4.

Exact leave-one-out predictive densities

For a fixed model M_g, under the Gaussian setting (4), the LOO-PD for each i, is

$$p(y(s_i) \mid \mathcal{D}_{-i}, M_g) = t_{2a_{*,i}}\left(y(s_i);\; J_i^\top V_{y_{-i}}^{-1} y_{-i} + A_i^\top B_i b_i,\; \frac{b_{*,i}}{a_{*,i}} (V_{y_i|y_{-i}} + A_i^\top B_i A_i) \right), \qquad (14)$$

where $J_i = R$`[-i, i]` is the cross-correlation matrix between $\{s_i\}$ and χ_{-i}, $(n-1) \times (n-1)$ matrix $V_{y_{-i}} = V_y$`[-i, -i]` is the marginal correlation matrix of y_{-i}, $B_i = X_{-i}^\top V_{y_{-i}}^{-1} X_{-i} + V_\beta^{-1}$, $b_i = X_{-i}^\top V_{y_{-i}}^{-1} y_{-i} + V_\beta^{-1} \mu_\beta$, $p \times 1$ vector $A_i = X_i - X_{-i}^\top V_{y_{-i}}^{-1} J_i$ with $X_i^\top = X$`[i, ]`, scalar $V_{y_i|y_{-i}} = V_{y_i} - J_i^\top V_{y_{-i}}^{-1} J_i$ with $V_{y_i} = V_y$`[i, i]`, $a_{*,i} = a_\sigma + (n-1)/2$, $b_{*,i} = b_\sigma + (y_{-i}^\top V_{y_{-i}}^{-1} y_{-i} + \mu_\beta^\top V_\beta^{-1} \mu_\beta - b_i^\top B_i b_i)/2$ and, $t_\rho(x; m, v^2)$ denotes the t-density with degrees of freedom ρ, location and scale parameters m and v respectively, evaluated at x. Note that, the computation surrounding (14) surrounds the Cholesky factor $L_{y_{-i}} =$ `chol`$(V_{y_{-i}})$, and subsequent triangular solves to obtain $L_{y_{-i}}^{-1} y_{-i}$, $L_{y_{-i}}^{-1} X_{-i}$ and, $L_{y_{-i}}^{-1} J_i$. Given that, we have already obtained L_y, the Cholesky factor of V_y while sampling from $p(\theta \mid \mathcal{D}, M_g)$, it is not necessary to evaluate `chol`$(V_{y_{-i}})$ again for each $i = 1, \ldots, n$. Instead, we do

```
for $i = 1, \ldots, n$ do
    $L_{y_{-i}} \leftarrow \texttt{cholDelUpdate}(L_y, i)$                    $\triangleright$ $O((n-i)^2)$ flops
    $L_{y_{-i}}^{-1} y_{-i} \leftarrow \texttt{trsolve}(L_{y_{-i}}, y_{-i})$          $\triangleright$ $O(n^2)$ flops
    $L_{y_{-i}}^{-1} X_{-i} \leftarrow \texttt{trsolve}(L_{y_{-i}}, X_{-i})$         $\triangleright$ $O(pn^2)$ flops
    $L_{y_{-i}}^{-1} J_i \leftarrow \texttt{trsolve}(L_{y_{-i}}, J_i)$              $\triangleright$ $O(n^2)$ flops
    Compute $p(y(s_i) \mid \mathcal{D}_{-i}, M)$ using the closed form expression in (14)
end for
```

where `cholDelUpdate`(L_y, i) refers to updating the Cholesky factor L_y after deleting the i-th row and column from V_y. This accumulates to a computational complexity of $O(n^3)$, a substantial improvement over the naive approach that requires $O(n^4)$ flops. Calculation of exact LOO-PD under a fixed model M_g within (4) can be accessed in the `spLMexact()` function by setting the optional arguments `loopd = TRUE` and `loopd.method = "exact"`.

Under the non-Gaussian setting (8), for a fixed model M_g, evaluation of LOO-PD is exceptionally challenging, since, unlike the Gaussian case, they do not have closed form expressions.

Instead, we use draws $\{\beta_{g,i}^{(\ell)}, z_{g,i}^{(\ell)}\}_{\ell=1}^{N_{\rm mc}}$ from $p(\theta \mid \mathcal{D}_{-i}, M_g)$ in (9), to evaluate

$$p(y(s_i) \mid \mathcal{D}_{-i}, M_g) = \frac{1}{N_{\rm mc}} \sum_{\ell=1}^{N_{\rm mc}} \mathrm{EF}\left(y(s_i);\, x(s_i)^\top \beta_{g,i}^{(\ell)} + z_{g,i}^{(\ell)}(s_i); b(s_i), \psi_y\right) , \tag{15}$$

where $N_{\rm mc}$ is for the number of Monte Carlo simulations for evaluating (15), and, $\mathrm{EF}(x; \eta; b, \psi)$ corresponds to the exponential family density with log partition function ψ, evaluated as x (see Appendix A for details). While it is possible to proceed by applying `cholDelUpdate()` to obtain row-deleted Cholesky factors, computation of the *priming* step (see Section 2.2.1), that is required to set up matrices for the projection (10) is of order $O(n^3)$ and hence, offsets the computational efficiency gained by `cholDelUpdate()`. This utility is offered in the `spGLMexact()` function invoked with arguments `loopd = TRUE` and `loopd.method = "exact"`.

K-fold cross validation

Following Vehtari, Gelman, and Gabry (2017), a more practical approach for evaluation LOO-PD is to cross-validate using $K \ll n$ hold-out partitions of $\mathcal{D}$. Without loss of generality, we partition the indices $\{1, \ldots, n\}$ into K blocks such that each block consists of consecutive indices $\mathcal{I}_k$ of size n_k, $k = 1, \ldots, K$. Let $\mathcal{D}_{[k]} = (y_{[k]}, X_{[k]}, \chi_{[k]})$ be the subset of $\mathcal{D}$ corresponding to the indices in $\mathcal{I}_k$, and $\mathcal{D}_{[-k]} = (y_{[-k]}, X_{[-k]}, \chi_{[-k]})$ is its complement i.e., $\mathcal{D} \setminus \mathcal{D}_{[k]}$ of size $(n - n_k)$, for each k. Under the Gaussian setting, such strategies can be applied to approximate $p(y(s_i) \mid \mathcal{D}_{-i}, M_g)$ by $p(y(s_i) \mid \mathcal{D}_{-[k]}, M_g)$ which replaces $L_{y_{-i}} \leftarrow \texttt{cholDelUpdate}(L_y, i)$ by,

$$L_{y_{-[k]}} \leftarrow \texttt{cholBlockDelUpdate}(L_y, \mathcal{I}_k) ,$$

where $\texttt{cholBlockDelUpdate}(L_y, \mathcal{I}_k)$ corresponds to the block-deletion update of the Cholesky factor L_y after removing the rows and columns pertaining to $\mathcal{I}_k$ from V_y. However, this does not add any computational advantage over the exact leave-one-out strategy since each of the K block-deletion updates would require n_k rank-one updates for $k = 1, \ldots, K$, which accumulates to a similar flop count as n single row-deletion updates.

On the other hand, under the non-Gaussian setting, due to the intractability of the LOO-PD, K-fold cross-validation can be very useful. For each k, we fit M_g to $\mathcal{D}_{-[k]}$ and draw $N_{\rm MC}$ samples $\{\beta_{g,k}^{(\ell)}, z_{g,k}^{(\ell)}\}_{\ell=1}^{N_{\rm mc}}$ from $p((\gamma^\top, q^\top)^\top \mid \mathcal{D}_{-[k]}, M_g)$ as given in (9), to evaluate

$$p(y(s_i) \mid \mathcal{D}_{-i}, M_g) \approx \frac{1}{N_{\rm mc}} \sum_{\ell=1}^{N_{\rm mc}} \mathrm{EF}\left(y(s_i);\, x(s_i)^\top \beta_{g,k}^{(\ell)} + z_{g,k}^{(\ell)}(s_i); b(s_i), \psi_y\right), \forall s_i \in \chi_{[k]} , \tag{16}$$

for locations pertaining to partition $\mathcal{D}_{[k]}$. This brings down the computational complexity to $O(Kn^3)$ flops. A typical value in the literature is $K = 10$ (see Vehtari *et al.* 2017, Section 2.3). For example, under a fixed model in the non-Gaussian setting , evaluation of LOO-PD using 10-fold cross-validation can be accessed within the `spGLMexact()` function by running it with the arguments `loopd = TRUE`, `loopd.method = "CV"` and `CV.K = 10`.

Pareto-smoothed importance sampling

Importance weighting has also been an attractive option to approximate leave-one-out predictive densities when the evaluation involves Monte Carlo integration (see Gelfand, Dey, and Chang 1992; Vehtari, Simpson, Gelman, Yao, and Gabry 2024b) as seen in Equations 15 and 16. Note that, under any model M_g, Gaussian or non-Gaussian, based on draws $\{\beta_g^{(\ell)}, z_g^{(\ell)}\}_{\ell=1}^{N_{\text{mc}}}$ from the full posterior $p(\theta \mid \mathcal{D}, M_g)$, we can approximate LOO-PD as

$$p(y(s_i) \mid \mathcal{D}_{-i}, M_g) \approx \frac{1}{\sum_{\ell=1}^{N_{\text{mc}}} w_{i,g}^{\ell}} \sum_{\ell=1}^{N_{\text{mc}}} w_{i,g}^{\ell} \,\text{EF}\left(y(s_i);\, x(s_i)^{\top}\beta_g^{(\ell)} + z_g^{(\ell)}(s_i); b(s_i), \psi_y\right) , \quad (17)$$

where $w_{i,g}^{\ell}$ is the importance ratio with $1/w_{i,g}^{\ell} = \text{EF}(y(s_i);\, x(s_i)^{\top}\beta_g^{(\ell)} + z_g^{(\ell)}(s_i); b(s_i), \psi_y)$. The weights $w_{i,g}^{\ell}$ tend to have high or infinite variance thus introducing instability in computing (17). To address this issues, Vehtari *et al.* (2017) proposes stabilizing the weights by fitting a generalized Pareto distribution to the tail of the weight distribution using the empirical Bayes estimation algorithm proposed in Zhang and Stephens (2009). Thus, no model fitting is necessary for calculating leave-one-out predictive densities. The computational cost of this approach is $O(n \log n)$ and hence very fast. We follow the R package **loo** for developing this functionality for or package. Currently, we implement this approach only for the Gaussian model, and can be accessed by setting the arguments `loopd = TRUE` and `loopd.method = "PSIS"` in the function `spLMexact()`.

4. Computational workflow

The algorithms for exact sampling from the posterior distributions and calculation leave-one-out predictive densities, as presented in the previous sections, are implemented in **spStack** functions. The backend of the package is implemented in C++ and harnesses R's *Foreign Language Interface* to call Fortran routines for optimized matrix computations. To be specific, we leverage the `F77_NAME` macro to interface with legacy Fortran functions offered by highly efficient linear algebra libraries like **BLAS** (Basic Linear Algebra Subprograms, Lawson, Hanson, Kincaid, and Krogh 1979) and **LAPACK** (Linear Algebra Package, Anderson, Bai, Bischof, Blackford, Demmel, Dongarra, Croz, Greenbaum, Hammarling, McKenney, and Sorensen 1999). The algorithms for exact posterior sampling and calculation of leave-one-out predictive densities, as presented in Sections 2 and 3, require expensive matrix operations and hence we place substantial effort on formulating algorithms in order to optimize floating point operations and allocations of dynamic memory. We organize our contribution towards enhancing computational efficiency into the following —

- Using suitable routines that are already existing in efficient linear algebra libraries like **BLAS** and **LAPACK**. In particular, we use level 2 **BLAS** routines like `dtrsv` for solving triangular systems $Ax = b$ where both x, b are vectors and A is a triangular matrix, `dgemv` for matrix-vector multiplication, `ddot` for computing dot product, and level 3 **BLAS** routines like `dtrsm` for solving $AX = b$ where both X, B are matrices and A is a triangular matrix, `dgemm` for matrix-matrix multiplication, etc. We also use the `dpotrf` routine from **LAPACK**, referred to as `chol()` in the previous sections, for computing Cholesky factors of dense matrices. `trsolve()` is used to refer to either `trsv` or `trsm`.

- Developing new routines that are not readily available within popular matrix algebra libraries, to execute our proposed algorithms. These include C++ functions for the utilities `cholDelUpdate()`, `cholBlockDelUpdate()` and `cholRankOneUpdate()`. The

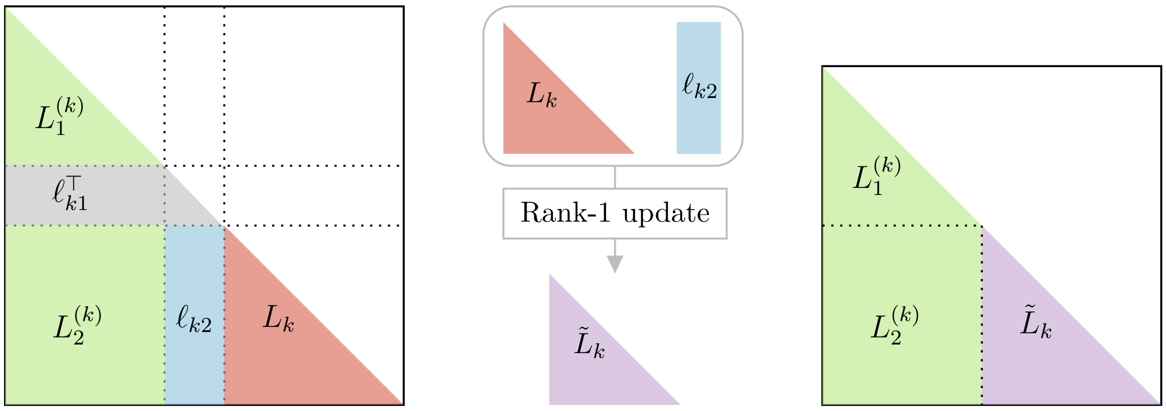

Figure 1: A schematic diagram summarizing the row-deletion update of a Cholesky factor

first two functions are defined in Section 3.2, and, `cholRankOneUpdate()` corresponds to the rank-1 update of a Cholesky factor. Details are discussed below.

The `cholDelUpdate`(L, k) refers to finding the Cholesky factor of the $(n-1)\times(n-1)$ matrix A_{-k} which is formed my deleting the k-th row and column from the $n \times n$ matrix A for some $1 < k < n$, where L and $\tilde{L}$ denote the lower-triangular Cholesky factors of A and A_{-k}, respectively. Note that, the k-th row of L partitions the factor L into five components, the top left $(k-1)\times(k-1)$ lower-triangular submatrix $L_1^{(k)}$, the non-zero $k \times 1$ submatrix of the k-th row $\ell_{k1}^\top$, the lower left $(n-k)\times(k-1)$ submatrix $L_2^{(k)}$, the non-zero $(n-k)\times 1$ submatrix of k-th column ℓ_{k2}, and, the bottom right $(n-k)\times(n-k)$ lower-triangular submatrix L_k. See the left diagram of Figure 1 for reference. Similarly, for $\tilde{L}$, let $\tilde{L}_1^{(k)}$ be the top left $(k-1)\times(k-1)$ submatrix, $\tilde{L}_2^{(k)}$ be the lower left $(n-k)\times(k-1)$ submatrix and $\tilde{L}_k$ be the lower right $(n-k)\times(n-k)$ submatrix. Then equating $(LL^\top)$`[-k, -k]` $= \tilde{L}\tilde{L}^\top$, we get

$$\tilde{L}_1^{(k)} \leftarrow L_1^{(k)}, \quad \tilde{L}_2^{(k)} \leftarrow L_2^{(k)}, \quad \tilde{L}^{(k)} \leftarrow \texttt{chol}(L_k L_k^\top + \ell_{k2}\ell_{k2}^\top) \, ,$$

where $\tilde{L}^{(k)}$ is simply a rank-1 update of L_k. Hence, `cholDelUpdate`(L, k) boils down to a simple rank-1 update of a $(n-k)\times(n-k)$ submatrix of the factor L. In case of `cholBlockDelUpdate`(L, $\mathcal{I}$), where $\mathcal{I}$ is a subset of the indices $\{1,\ldots,n\}$, of size k, and consisting of consecutive indices, the problem simply reduces to a rank-k update of a $(n-m)\times(n-m)$ matrix where $m = \max(\mathcal{I})$. While the R package **mgcv** provide these utilities for rank-1 updates following Golub and Van Loan (2013), we implement a more memory-efficient algorithm proposed by Krause and Igel (2015). In **spStack**, we provide a R wrapper for all these C++ routines in the R functions `cholUpdateRankOne()`, `cholUpdateDel()` and `cholUpdateDelBlock()` to provide utilities for advanced practitioners interested in developing their own cross-validation algorithms. See examples in Appendix B.

spStack also depends on some R packages including **CVXR** which is used to find optimal stacking weights, **future** (Bengtsson 2021) and **future.apply** to utilize its cross-platform parallelization plans to access multi-core implementation of our stacking algorithm. To be specific, depending on parallel backend available, users can set up their own plan using the `future()` function and simply set the argument `parallel = TRUE` in the functions `spLMstack()` or, `spGLMstack()` to execute posterior sampling for the G candidate models in parallel. Other imports include **ggplot2** and **MBA** that are used mainly for visualization purposes.

5. Illustrations

In this section, we demonstrate different models offered by **spStack** using synthetic data.

5.1. Gaussian spatial regression

Section 2.1 provides code for sampling from the posterior (5) conditional on fixed values of θ_{sp} and δ^2 using the function `spLMexact()`. This function also has optional argument `loopd`, which when set to `TRUE`, evaluates leave-one-out predictive densities using the method assigned by the argument `loopd.method`. Valid arguments for `loop.method` are `"exact"` which evaluates (14), and `"PSIS"` which uses (17). See the following code.

```
R> set.seed(1729)
R> mod3 <- spLMexact(y ~ x1, data = simGaussian,
+                     coords = as.matrix(simGaussian[, c("s1", "s2")]),
+                     cor.fn = "matern", spParams = list(phi = 3, nu = 0.75),
+                     noise_sp_ratio = 0.8, n.samples = 100,
+                     loopd = TRUE, loopd.method = "exact", verbose = FALSE)

R> mod4 <- spLMexact(y ~ x1, data = simGaussian,
+                     coords = as.matrix(simGaussian[, c("s1", "s2")]),
+                     cor.fn = "matern", spParams = list(phi = 3, nu = 0.75),
+                     noise_sp_ratio = 0.8, n.samples = 100,
+                     loopd = TRUE, loopd.method = "PSIS", verbose = FALSE)
```

We collect the LOO-PD calculated by these two methods and compare them in Figure 2.

```
R> loopd_exact <- mod3$loopd
R> loopd_psis <- mod4$loopd
R> loopd_df <- data.frame(exact = loopd_exact, psis = loopd_psis)
R> library(ggplot2)
R> ggplot(data = loopd_df, aes(x = exact)) +
+    geom_point(aes(y = psis), size = 1, alpha = 0.5) +
+    geom_abline(slope = 1, intercept = 0, color = "red", alpha = 0.5) +
+    xlab("Exact") + ylab("PSIS") + theme_bw() +
+    theme(panel.background = element_blank(), panel.grid = element_blank(),
+          aspect.ratio = 1)
```

From Figure 2, we observe that the approximation using PSIS works satisfactorily in terms of calculation of LOO-PD. Moreover, on a machine running Apple M1 chip with 8 GB memory, the median execution time for evaluation of LOO-PD is 2.5 seconds for exact and 350 milliseconds for PSIS, delivering a speed-up factor of approximately 7.5.

Next, we illustrate the `spLMstack()` function for implementing Bayesian predictive stacking algorithm for spatial Gaussian data. We use the default settings for the priors, choose the LOO-PD calculation method to be PSIS, and set some candidate values of θ_{sp} and δ^2. The set of candidate models are constructed by taking a Cartesian product of the grid of candidate values of the parameters. For example, in the following code, we have 3 candidate values for ϕ, 2 candidate values each for ν and δ^2, which entails $3 \times 2 \times 2 = 12$ candidate models.

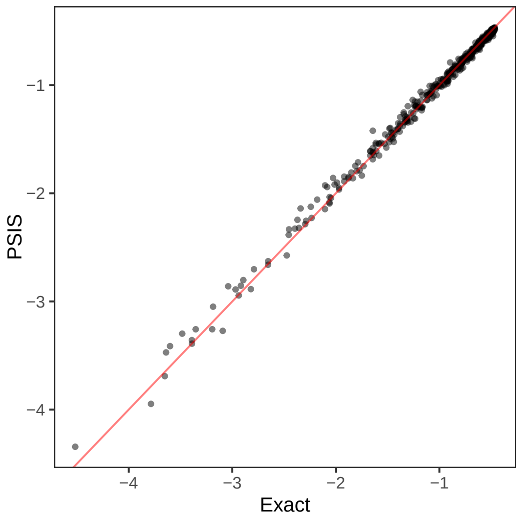

Figure 2: Comparison of exact leave-one-out predictive densities with that calculated using Pareto-smoothed importance sampling.

```
R> mod5 <- spLMstack(y ~ x1, data = simGaussian,
+                    coords = as.matrix(simGaussian[, c("s1", "s2")]),
+                    cor.fn = "matern",
+                    params.list = list(phi = c(1.5, 3, 5),
+                                       nu = c(0.5, 1.5),
+                                       noise_sp_ratio = c(0.5, 1.5)),
+                    n.samples = 1000, loopd.method = "PSIS",
+                    parallel = FALSE, solver = "ECOS", verbose = TRUE)
```

```
STACKING WEIGHTS:

           | phi | nu  | noise_sp_ratio | weight |
+----------+-----+-----+----------------+--------+
| Model 1  |  1.5|  0.5|             0.5| 0.000  |
| Model 2  |  3.0|  0.5|             0.5| 0.148  |
| Model 3  |  5.0|  0.5|             0.5| 0.000  |
| Model 4  |  1.5|  1.5|             0.5| 0.000  |
| Model 5  |  3.0|  1.5|             0.5| 0.000  |
| Model 6  |  5.0|  1.5|             0.5| 0.852  |
| Model 7  |  1.5|  0.5|             1.5| 0.000  |
| Model 8  |  3.0|  0.5|             1.5| 0.000  |
| Model 9  |  5.0|  0.5|             1.5| 0.000  |
| Model 10 |  1.5|  1.5|             1.5| 0.000  |
| Model 11 |  3.0|  1.5|             1.5| 0.000  |
| Model 12 |  5.0|  1.5|             1.5| 0.000  |
+----------+-----+-----+----------------+--------+
```

Users can check the solver status of the optimization by issuing the following.

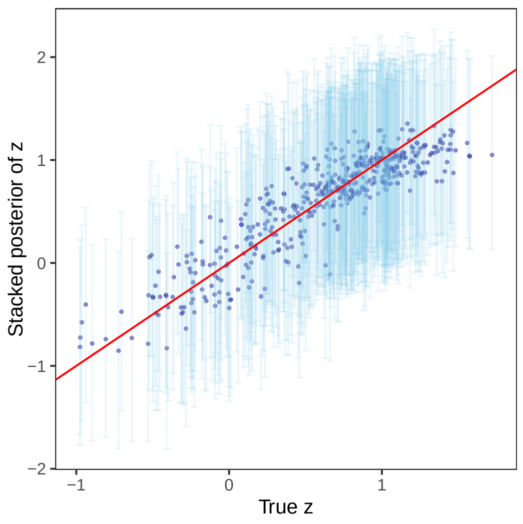

Figure 3: Comparison of true spatial effects and their corresponding stacked posterior distributions in analysis of synthetic Gaussian data.

```
R> print(mod5[["solver.status"]])

"optimal"
```

spStack also provides the helper function called `stackedSampler()` to sample from the *stacked* posterior. Subsequent posterior inference proceeds from these samples.

```
R> post_samps <- stackedSampler(mod5)
R> post_beta <- post_samps[["beta"]]
R> summary_beta <- t(apply(post_beta, 1,
+                           function(x) quantile(x, c(0.025, 0.5, 0.975))))
R> rownames(summary_beta) <- mod5[["X.names"]]
R> print(summary_beta)

                2.5%      50%    97.5%
(Intercept) 1.185252 2.040483 2.873048
x1          4.949013 5.001442 5.052616
```

Moreover, we compare the posterior samples of the spatial random effects with their corresponding true values in Figure 3. **spStack** also provides functions to plot interpolated spatial surfaces for visualization purposes. While `surfaceplot()` creates a single spatial surface plot, `surfaceplot2()` returns two side-by-side surface plots with a common color scale.

```
R> postmedian_z <- apply(post_z, 1, median)
R> simGaussian$z_hat <- postmedian_z
R> plot_z <- surfaceplot2(simGaussian, coords_name = c("s1", "s2"),
+                         var1_name = "z_true", var2_name = "z_hat")
R> library(ggpubr)
R> ggarrange(plotlist = plot_z, common.legend = TRUE, legend = "right")
```

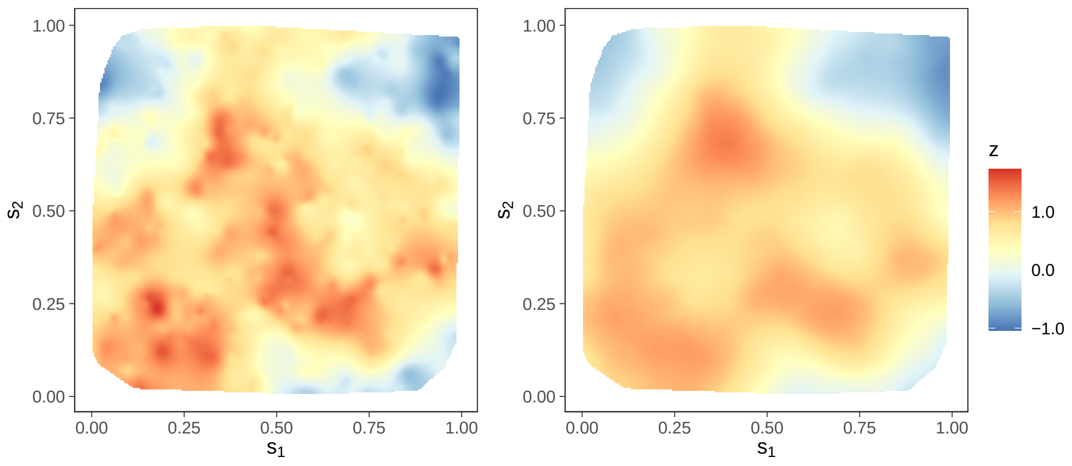

Figure 4: Comparison of interpolated spatial surfaces of true spatial effects (left) and their corresponding stacked posterior median (right) in analysis of synthetic Gaussian data.

5.2. Non-Gaussian spatial regression

In **spStack**, we offer functions for Bayesian analysis of geostatistical count data including Poisson and binomial counts as well as spatial binary data. Section 2.2 introduces the function `spGLMeaxct()` which facilitates drawing samples from (9) conditional on fixed values of process parameters θ_{sp} and boundary adjustment parameter α_ϵ. For implementing our stacking algorithm, we build our collection of candidate models based on a Cartesian product of a grid of candidate values of these parameters. This can be supplied into `spGLMstack()` using the argument `params.list` which is a named list of maximum length 3. The list contains candidate values of `phi`, `nu` and `boundary`, with each containing a numeric vector of candidate values of ϕ, ν and α_ϵ, respectively. The arguments `n.samples` denote the number of posterior samples N_s, and, `loopd.controls` is a list containing different parameters necessary to substantiate the calculation of LOO-PD. For example, `loopd.controls = list(method = "CV", CV.K = 10, nMC = 1000)` corresponds to 10-fold cross validation with number of Monte Carlo samples $N_{\text{mc}} = 1000$. See below for examples.

Analysis of spatial poisson counts

In practice, point-referenced count data are often modeled as poisson counts, e.g., tree counts in forestry, species sightings in ecology, traffic accident counts etc. We demonstrate below how to use `spGLMexact()` to analyze such data. Suppose we choose 3 candidate values of ϕ and 2 candidate values each for ν and α_ϵ, and store them in `cand_list`. Note that, this implies 12 candidate models in total. In addition, we also assign the settings under which the LOO-PD will be calculated in `loopd_settings`.

```
R> cand_list <- list(phi = c(3, 7, 10),
+                    nu = c(0.5, 1.5),
+                    boundary = c(0.5, 0.6))
R> loopd_settings <- list(method = "CV", CV.K = 10, nMC = 1000)
```

Here, we show how to use the **future** package to implement `spGLMstack()` in parallel. Issuing `plan("multicore", workers = 6)` divides the task of fitting 12 models into 6 cores. After

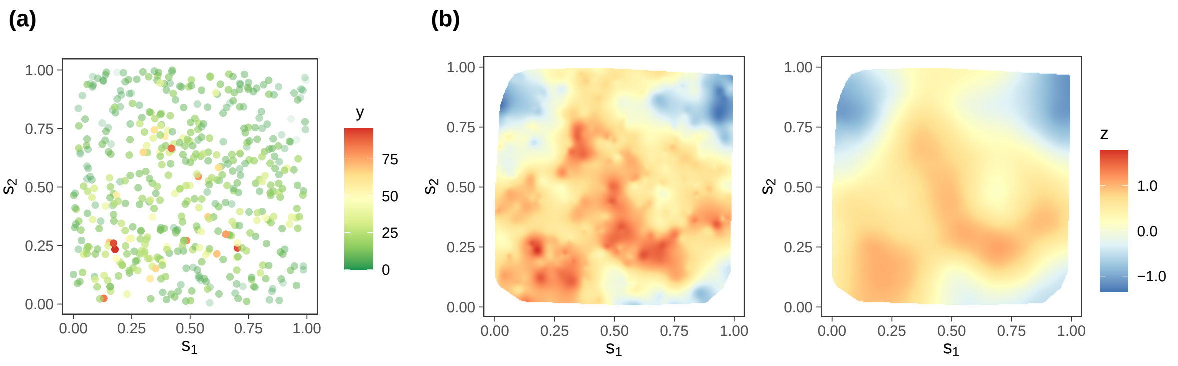

Figure 5: Analysis of synthetic spatial poisson counts: (a) Raw data, (b) interpolated spatial surfaces of true spatial effects (left) and their corresponding posterior median (right).

completion of the task, we close the forking process by issuing `plan("sequential")`. See the **future** manual for custom parallelization plans on systems with advanced parallel backends.

```
R> library("future")
R> plan("multicore", workers = 6)
R> set.seed(1729)
R> mod6 <- spGLMstack(y ~ x1, data = simPoisson, family = "poisson",
+                     coords = as.matrix(simPoisson[, c("s1", "s2")]),
+                     cor.fn = "matern", params.list = cand_list,
+                     n.samples = 1000, loopd.controls = loopd_settings,
+                     parallel = FALSE, solver = "ECOS", verbose = TRUE)
R> plan("sequential")
```

Similar to `spLMstack()`, `spGLMstack()` also returns an object of class '`spGLMstack`', which can be passed into `stackedSampler()` for obtaining samples from the stacked posterior distribution. The samples from the stacked posterior can be analyzed as discussed earlier. Figure 5 shows the raw spatial counts and the interpolated spatial surfaces of the true spatial effects and their posterior median based on samples drawn from the stacked posterior. Figure 5 compares the raw counts and the learning of the spatial surface. We observe elevated values on spatial surface of the posterior median of the spatial effects in regions with higher counts and successfully captures the pattern seen in the true spatial surface.

Analysis of spatial binomial counts

Point-referenced binomial counts is a common choice of model in surveys studying prevalence of various diseases over large regions. For example, Wong, Flegg, Golding, and Kandanaarachchi (2023) provides an elaborate review of recent computational methods for geostatistical modeling of spatial binomial count data to map malaria prevalence. The syntax for analyzing binomial counts using `spGLMexact()` is very similar with a small change in the symbolic formula that defines the model. For example, in the synthetic binomial count data `simBinom` that is available in **spStack**, besides the predictor `x1`, it has the response `y` which denotes the number of successes and `n_trials` which denotes the total number of trials at each location. We express the model using the symbolic formula `cbind(y, n_trials) ~ x1`.

```
R> data(simBinom)
R> cand_list <- list(phi = c(2, 5, 8), nu = c(1.0, 1.5),
+                     boundary = c(0.5, 0.6))
R> mod7 <- spGLMstack(cbind(y, n_trials) ~ x1, data = simBinom,
+                      family = "binomial",
+                      coords = as.matrix(simBinom[, c("s1", "s2")]),
+                      cor.fn = "matern", params.list = cand_list,
+                      n.samples = 1000, loopd.controls = loopd_settings,
+                      parallel = FALSE, solver = "ECOS", verbose = TRUE)
```

Analysis of spatial binary data

Point-referenced spatial binary data (e.g., presence/absence) is widely used in many scientific disciplines including, but not limited to, satellite imagery, species occurrence, voting outcomes etc. See below example code for analyzing such data using the `spGLMexact()` function.

```
R> data(simBinary)
R> cand_list <- list(phi = c(2, 5, 8), nu = c(0.75, 1.25),
+                     boundary = c(0.5, 0.6))
R> mod8 <- spGLMstack(y ~ x1, data = simBinary, family = "binary",
+                      coords = as.matrix(simBinom[, c("s1", "s2")]),
+                      cor.fn = "matern", params.list = cand_list,
+                      n.samples = 1000, loopd.controls = loopd_settings,
+                      parallel = FALSE, solver = "ECOS", verbose = TRUE)
```

This returns an object of class '`spGLMstack`' and can be analyzed similarly, as discussed earlier in case of the synthetic spatial poisson count data.

6. Summary and future directions

The development of **spStack** is aimed towards delivering user-friendly functions for Bayesian analysis of geostatistical data. Since traditional Bayesian geostatistical models typically involve repeated expensive dense matrix operations within MCMC and hence entails unreasonably prolonged execution times, **spStack** enhances the practicality of such models by demonstrating Bayesian predictive stacking to be an effective tool for estimating spatial regression models and yielding robust predictions for both Gaussian and non-Gaussian point-referenced spatial data. We exploit analytically accessible distribution theory pertaining to Bayesian analysis of linear mixed models (LMM) as well as generalized linear mixed models (GLMM), that enables us to directly sample from the posterior distributions. From a computational viewpoint, **spStack** implements efficient algorithms to tackle the non-trivial and challenging problem of evaluating leave-one-out predictive densities for spatially dependent data. Core functions also features seamless integration with different parallel backends further accelerating inference. The contribution of **spStack** falls under integrative Bayesian learning as it focuses largely on effectively combining inference across different closed-form posterior distributions of geostatistical models by circumventing inference on weakly identified parameters. Future developments will consider functionalities for analyzing different kinds of spatial-temporal data, and, adapting to variants of Gaussian process models that scale inference to massive datasets by evading the Cholesky decomposition of dense correlation matrices.

Computational details

The results in this paper were obtained using R 4.4.1 with programs executed on platform aarch64-apple-darwin20 with an Apple silicon M1 chip, running under MacOS Sequoia 15.0.1.

Package versions: **spStack** 1.0.1, **future** 1.34.0, **ggpubr** 0.6.0, **ggplot2** 3.5.1, and **knitr** 1.48.

Acknowledgments

Sudipto Banerjee and Soumyakanti Pan were supported by two research grants from the National Institute of Environmental Health Sciences (NIEHS), one grant from the National Institute of General Medical Science (NIGMS) and another from the Division of Mathematical Sciences of the National Science Foundation (NSF-DMS).

References

Abramowitz M, Stegun IA (eds.) (1965). *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables.* Dover Publications, Inc., New York.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Croz JD, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999). ***LAPACK** Users' Guide.* Philadelphia, PA, third edition. doi:10.1137/1.9780898719604. URL http://www.netlib.org/lapack/lug/.

ApS M (2024). ***Rmosek**: The R-to-**MOSEK** optimization interface.* R package version 10.2.0, URL http://www.mosek.com/.

Bakar KS, Sahu SK (2015). "**spTimer**: Spatio-Temporal Bayesian Modeling Using R." *Journal of Statistical Software*, **63**(15), 1–32. doi:10.18637/jss.v063.i15. URL https://doi.org/10.18637/jss.v063.i15.

Banerjee S (2020). "Modeling massive spatial datasets using a conjugate Bayesian linear modeling framework." *Spatial Statistics*, **37**, 100417. ISSN 2211-6753. doi:https://doi.org/10.1016/j.spasta.2020.100417. Frontiers in Spatial and Spatio-temporal Research.

Banerjee S, Carlin BP, Gelfand AE (2014). *Hierarchical Modeling and Analysis for Spatial Data.* 2nd edition. Chapman and Hall/CRC, New York. doi:10.1201/b17115.

Bengtsson H (2021). "A Unifying Framework for Parallel and Distributed Processing in R using Futures." *The R Journal*, **13**(2), 208–227. doi:10.32614/RJ-2021-048. URL https://doi.org/10.32614/RJ-2021-048.

Bernardo J, Smith A (1994). *Bayesian Theory.* John Wiley and Sons, Chichester, UK.

Bivand R, Nowosad J (2024). *CRAN Task View: Analysis of Spatial Data*. Version 2024-06-18, URL https://CRAN.R-project.org/view=Spatial.

Bradley JR, Clinch M (2024). "Generating Independent Replicates Directly from the Posterior Distribution for a Class of Spatial Hierarchical Models." *Journal of Computational and Graphical Statistics*, **0**(0), 1–17. doi:10.1080/10618600.2024.2365728.

Bradley JR, Holan SH, Wikle CK (2020). "Bayesian Hierarchical Models With Conjugate Full-Conditional Distributions for Dependent Data From the Natural Exponential Family." *Journal of the American Statistical Association*, **115**(532), 2037–2052. doi:10.1080/01621459.2019.1677471.

Breiman L (1996). "Stacked regressions." *Machine learning*, **24**(1), 49–64. doi:10.1023/A:1018046112532.

Carlin BP, Louis TA (2008). *Bayesian Methods for Data Analysis*. 3rd edition. Chapman and Hall/CRC, New York. doi:10.1201/b14884.

Chilès JP, Delfiner P (2012). *Geostatistics: Modeling Spatial Uncertainty*. John Wiley & Sons, New York. doi:10.1002/9781118136188.

Christensen OF, Ribeiro PJ (2002). "**geoRglm**: A package for generalised linear spatial models." *R News*, **2**, 26–28. ISSN 1609-3631. URL https://journal.r-project.org/articles/RN-2002-013/.

Clyde MA, Iversen ES (2013). "Bayesian model averaging in the M-open framework." In *Bayesian Theory and Applications*, pp. 483–498. Oxford University Press. ISBN 9780199695607. doi:10.1093/acprof:oso/9780199695607.003.0024.

Cressie NAC (1993). *Statistics for Spatial Data*. 2nd edition. John Wiley & Sons, New York. doi:10.1002/9781119115151.

de Valpine P, Turek D, Paciorek C, Anderson-Bergman C, Temple Lang D, Bodik R (2017). "Programming with models: writing statistical algorithms for general model structures with NIMBLE." *Journal of Computational and Graphical Statistics*, **26**, 403–417. doi:10.1080/10618600.2016.1172487.

Diaconis P, Ylvisaker D (1979). "Conjugate Priors for Exponential Families." *The Annals of Statistics*, **7**(2), 269 – 281. doi:10.1214/aos/1176344611.

Diggle PJ, Ribeiro PJ (2007). *Model-based Geostatistics*. Springer-Verlag New York, New York. doi:10.1007/978-0-387-48536-2.

Ding P (2016). "On the Conditional Distribution of the Multivariate t Distribution." *The American Statistician*, **70**(3), 293–295. doi:10.1080/00031305.2016.1164756.

Finley AO, Banerjee S, Carlin BP (2007). "**spBayes**: An R Package for Univariate and Multivariate Hierarchical Point-referenced Spatial Models." *Journal of Statistical Software*, **19**(4), 1–24. doi:10.18637/jss.v019.i04.

Fu A, Narasimhan B, Boyd S (2020). "**CVXR**: An R Package for Disciplined Convex Optimization." *Journal of Statistical Software*, **94**(14), 1–34. doi:10.18637/jss.v094.i14. URL https://www.jstatsoft.org/index.php/jss/article/view/v094i14.

Gelfand AE, Dey DK, Chang H (1992). "Model Determination using Predictive Distributions with Implementation via Sampling-Based Methods." In *Bayesian Statistics 4: Proceedings of the Fourth Valencia International Meeting, Dedicated to the memory of Morris H. DeGroot, 1931–1989*. Oxford University Press. ISBN 9780198522669. doi:10.1093/oso/9780198522669.003.0009.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian Data Analysis*. 3rd edition. Chapman and Hall/CRC, New York. doi:10.1201/b16018.

Gilks W, Richardson S, Spiegelhalter D (1995). *Markov Chain Monte Carlo in Practice*. 1st edition. Chapman and Hall/CRC, New York. doi:10.1201/b14835.

Golub GH, Van Loan CF (2013). *Matrix Computations - 4th Edition*. 4th edition. Johns Hopkins University Press, Philadelphia, PA. doi:10.1137/1.9781421407944.

Hoeting JA, Madigan D, Raftery AE, Volinsky CT (1999). "Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and E. I. George, and a rejoinder by the authors)." *Statistical Science*, **14**(4), 382 – 417. doi:10.1214/ss/1009212519.

Krause O, Igel C (2015). "A More Efficient Rank-one Covariance Matrix Update for Evolution Strategies." In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, p. 129–136. Association for Computing Machinery, New York, NY, USA. ISBN 9781450334341. doi:10.1145/2725494.2725496.

Lawson C, Hanson R, Kincaid D, Krogh F (1979). "Basic Linear Algebra Subprograms for Fortran usage." *ACM Transactions on Mathematical Software*, **5**(3), 308–323. doi:10.1145/355841.355847.

Le T, Clarke B (2017). "A Bayes Interpretation of Stacking for $\mathcal{M}$-Complete and $\mathcal{M}$-Open Settings." *Bayesian Analysis*, **12**(3), 807 – 829. doi:10.1214/16-BA1023.

Lunn D, Spiegelhalter D, Thomas A, Best N (2009). "The BUGS project: Evolution, critique and future directions." *Statistics in Medicine*, **28**(25), 3049–3067. doi:https://doi.org/10.1002/sim.3680.

Madigan D, Raftery AE, Volinsky C, Hoeting J (1996). "Bayesian model averaging." In *Proceedings of the AAAI Workshop on Integrating Multiple Learned Models, Portland, OR*, pp. 77–83.

Moller J, Waagepetersen RP (2003). *Statistical Inference and Simulation for Spatial Point Processes*. Taylor & Francis, New York. doi:10.1201/9780203496930.

Neal R (2011). "MCMC Using Hamiltonian Dynamics." In *Handbook of Markov Chain Monte Carlo*, pp. 113–162. Chapman and Hall/CRC. doi:10.1201/b10905.

Pan S, Banerjee S (2024). ***spStack**: Bayesian Geostatistics Using Predictive Stacking*. R package version 1.0.1, URL https://CRAN.R-project.org/package=spStack.

Pan S, Zhang L, Bradley JR, Banerjee S (2024). "Bayesian Inference for Spatial-temporal Non-Gaussian Data Using Predictive Stacking." doi:10.48550/arXiv.2406.04655.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ribeiro Jr PJ, Diggle P, Christensen O, Schlather M, Bivand R, Ripley B (2024). ***geoR**: Analysis of Geostatistical Data*. R package version 1.9-4, URL https://CRAN.R-project.org/package=geoR.

Robert CP, Casella G (2004). *Monte Carlo Statistical Methods*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-1-4757-4145-2.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392. doi:https://doi.org/10.1111/j.1467-9868.2008.00700.x.

Sainsbury-Dale M, Zammit-Mangion A, Cressie N (2024). "Modeling Big, Heterogeneous, Non-Gaussian Spatial and Spatio-Temporal Data Using **FRK**." *Journal of Statistical Software*, **108**(10), 1–39. doi:10.18637/jss.v108.i10.

Schabenberger O, Gotway CA (2005). *Statistical Methods for Spatial Data Analysis*. Taylor & Francis, New York. doi:10.1201/9781315275086.

Smith BJ, Yan J, Cowles MK (2008). "Unified geostatistical modeling for data fusion and spatial heteroskedasticity with R package **ramps**." *Journal of Statistical Software*, **25**(10), 1–21. doi:10.18637/jss.v025.i10.

Stan Development Team (2024). *Stan Modeling Language Users Guide and Reference Manual*. Version 2.35, URL https://mc-stan.org.

Tae Yoon Kim JSP, Cox DD (2002). "Fast Algorithm for Cross-Validation of the Best Linear Unbiased Predictor." *Journal of Computational and Graphical Statistics*, **11**(4), 823–835. doi:10.1198/106186002826.

Thomas A, Best N, Lunn D, Arnold R, Spiegelhalter D (2014). ***GeoBUGS** User Manual*. Version 3.2.3, URL https://chjackson.github.io/openbugsdoc/GeoBUGS/Manuals/Manual.html.

Thomas A, O'Hara B, Ligges U, Sturtz S (2006). "Making BUGS open." *R News*, **6**(1), 12–17. URL https://cran.r-project.org/doc/Rnews/Rnews_2006-1.pdf.

Vehtari A, Gabry J, Magnusson M, Yao Y, Bürkner PC, Paananen T, Gelman A (2024a). "**loo**: Efficient leave-one-out cross-validation and WAIC for Bayesian models." R package version 2.8.0, URL https://mc-stan.org/loo/.

Vehtari A, Gelman A, Gabry J (2017). "Practical Bayesian Model Evaluation Using Leave-One-out Cross-Validation and WAIC." *Statistics and Computing*, **27**(5), 1413–1432. ISSN 0960-3174. doi:10.1007/s11222-016-9696-4.

Vehtari A, Simpson D, Gelman A, Yao Y, Gabry J (2024b). "Pareto Smoothed Importance Sampling." *Journal of Machine Learning Research*, **25**(72), 1–58. URL http://jmlr.org/papers/v25/19-556.html.

Wackernagel H (2003). *Multivariate Geostatistics.* Springer-Verlag Berlin, Heidelberg. doi:10.1007/978-3-662-05294-5.

Wikle CK, Cressie N (2011). *Statistics for Spatio-Temporal Data.* John Wiley & Sons.

Wolpert DH (1992). "Stacked generalization." *Neural Networks*, **5**(2), 241–259. ISSN 0893-6080. doi:https://doi.org/10.1016/S0893-6080(05)80023-1.

Wong S, Flegg JA, Golding N, Kandanaarachchi S (2023). "Comparison of new computational methods for spatial modelling of malaria." *Malaria Journal*, **22**(356). doi:10.1186/s12936-023-04760-7.

Yao Y, Pirš G, Vehtari A, Gelman A (2021). "Bayesian hierarchical stacking: Some models are (somewhere) useful." *Bayesian Analysis*, **1**(1), 1–29. doi:10.1214/21-BA1287.

Yao Y, Vehtari A, Gelman A (2022). "Stacking for Non-mixing Bayesian Computations: The Curse and Blessing of Multimodal Posteriors." *Journal of Machine Learning Research*, **23**(79), 1–45. URL http://jmlr.org/papers/v23/20-1426.html.

Yao Y, Vehtari A, Simpson D, Gelman A (2018). "Using Stacking to Average Bayesian Predictive Distributions (with Discussion)." *Bayesian Analysis*, **13**(3), 917 – 1007. doi:10.1214/17-BA1091.

Zammit-Mangion A, Cressie N (2021). "**FRK**: An R Package for Spatial and Spatio-Temporal Prediction with Large Datasets." *Journal of Statistical Software*, **98**(4), 1–48. doi:10.18637/jss.v098.i04.

Zhang H (2004). "Inconsistent Estimation and Asymptotically Equal Interpolations in Model-Based Geostatistics." *Journal of the American Statistical Association*, **99**(465), 250–261. doi:10.1198/016214504000000241.

Zhang J, Stephens MA (2009). "A New and Efficient Estimation Method for the Generalized Pareto Distribution." *Technometrics*, **51**(3), 316–325. doi:10.1198/tech.2009.08017.

Zhang L, Tang W, Banerjee S (2024). "Bayesian Geostatistics Using Predictive Stacking." doi:10.48550/arXiv.2304.12414.

A. Conjugate priors for exponential family

This section familiarizes the conjugate priors for the natural exponential family (Diaconis and Ylvisaker 1979) and their multivariate extensions (Bradley *et al.* 2020; Bradley and Clinch 2024). This family of distributions play a central role in development of the Bayesian hierarchical models for non-Gaussian outcomes.

A.1. The Diaconis-Ylvisaker distribution

Let Y be distributed from the natural exponential family, $\mathrm{EF}(\eta; b, \psi)$, with density

$$p(Y \mid \eta) = \exp\{\eta Y - b\psi(\eta) + c(Y)\},\ Y \in \mathscr{Y}, \eta \in \mathscr{H}\ , \tag{A1}$$

where $\mathscr{Y}$ denotes the support of Y and $\mathscr{H} = \{\eta : \psi(\eta) < \infty\}$ denotes the natural parameter space and, therefore, the support of η. The scalar b may be unknown, while $\psi(\cdot)$ and $c(\cdot)$ are known functions. Diaconis and Ylvisaker (1979) provides a proper conjugate prior for η in (A1) as $\mathrm{DY}(\alpha, \kappa; \psi)$, which has the density

$$p(\eta \mid \alpha, \kappa) \propto \exp\{\alpha\eta - \kappa\psi(\eta)\},\ \eta \in \mathscr{H}, \frac{\alpha}{\kappa} \in \mathscr{Y}, \kappa > 0\ . \tag{A2}$$

It is easily seen that the posterior distribution $\eta \mid Y, \alpha, \kappa \sim \mathrm{DY}(\alpha + Y, \kappa + b; \psi)$. There are several special cases of the DY distribution other than the Gaussian ($\psi = \psi_1$), log-gamma ($\psi = \psi_2$), and logit-beta ($\psi = \psi_3$) distributions, several of which do not correspond to a member of the exponential family. For example, $\alpha = 0$, $\psi(t) = \psi_4(t; \rho) = \log(1 + t^2/\rho)$, and $\kappa = (\rho + 1)/2$ with $\rho > 0$ results in a Student's t-distribution with ρ degrees of freedom.

A.2. The (Generalized) Conjugate Multivariate distribution

A multivariate version of (A2) is constructed using linear combinations of mutually independent DY random variables. Let ζ be the $n \times 1$ random vector

$$\zeta = \mu + L\eta\ , \tag{A3}$$

where $\zeta \in \mathscr{M}$, $\mathscr{M} = \{\zeta : \zeta = \mu + L\eta, \eta \in \mathscr{H}^n\}$, $\mu \in \mathbb{R}^n$ denotes a location vector, L is an $n \times n$ lower-triangular matrix with positive diagonal elements and the $n \times 1$ random vector $\eta = (\eta_1, \dots, \eta_n)^\top$ consists of n mutually independent DY random variables, $\eta_i \sim \mathrm{DY}(\alpha_i, \kappa_i; \psi)$ with $\kappa_i > 0$ for $i = 1, \dots, n$. Define $\zeta \sim \mathrm{CM}(\mu, L, \alpha, \kappa; \psi)$ with unnormalized density

$$p(\zeta \mid \mu, L, \alpha, \kappa) \propto \exp\left\{\alpha^\top L^{-1}(\zeta - \mu) - \kappa^\top \psi(L^{-1}(\zeta - \mu))\right\} \det(L^{-1}) \tag{A4}$$

for all $\zeta \in \mathscr{M}$, where ψ operates element-wise on $L^{-1}(\zeta - \mu)$, $\alpha = (\alpha_1, \dots, \alpha_n)^\top$ and $\kappa = (\kappa_1, \dots, \kappa_n)^\top$. If $\zeta = (\zeta_1^\top, \zeta_2^\top)^\top$ is distributed as $\mathrm{CM}(\mu, L, \alpha, \kappa; \psi)$, where ζ_1 is $r \times 1$ and ζ_2 is $(n - r) \times 1$, then the conditional distribution of ζ_1 given ζ_2 is $\mathrm{CM_c}(\mu^*, A_1, \alpha, \kappa; \psi)$ with

$$p(\zeta_1 \mid \zeta_2 = c_2, \mu^*, A_1, \alpha, \kappa) \propto \exp\{\alpha^\top (A_1\zeta_1 - \mu^*) - \kappa^\top \psi(A_1\zeta_1 - \mu^*)\}\ , \tag{A5}$$

as the unnormalized density for all $(\zeta_1^\top, c_2^\top)^\top \in \mathscr{M}$. Here, A_1 is defined as the $n \times r$ submatrix of $L^{-1} = [A_1 : A_2]$, and $\mu^* = L^{-1}\mu - A_2c_2$ for some $c_2 \in \mathbb{R}^{n-r}$. The proportionality constant in (A5) is strictly positive and finite ensuring that (A5) is proper (Bradley *et al.* 2020).

Bradley and Clinch (2024) generalize the CM distribution by relating ζ, μ, L and η as in (A3), where $\zeta = (\zeta_1^\top, \ldots, \zeta_K^\top)^\top$ and $\eta = (\eta_1^\top, \ldots, \eta_K^\top)^\top$ are $n \times 1$ with $n = \sum_{k=1}^K n_k$, and each element of η_k is independently distributed as $\eta_{k,i} \sim \mathrm{DY}(\alpha_{k,i}, \kappa_{k,i}; \psi_k)$, L is an $n \times n$ lower-triangular matrix with positive diagonal elements and μ is an $n \times 1$ location parameter. The density is written analogously to (A4) with $\zeta \in \mathscr{N}$, $\mathscr{N} = \{\zeta : \zeta = \mu + L\eta, \eta_{k,i} \in \mathscr{H}_k, i = 1, \ldots, n_k, k = 1, \ldots, K\}$ with $\mathscr{H}_k$ being the parameter space corresponding to the log partition function ψ_k, $\alpha_{k,i}/\kappa_{k,i} \in \mathscr{Y}_k$, $\kappa_{k,i} > 0$ and $\psi(L^{-1}(\zeta - \mu)) = (\psi_1(J_1 L^{-1}(\zeta - \mu))^\top, \ldots, \psi_K(J_K L^{-1}(\zeta - \mu))^\top)^\top$, where $J_k = [0 : I_{n_k} : 0]$ is $n_k \times n$ and each $\psi_k(\cdot)$ operates element-wise on the vector of arguments, $\alpha = (\alpha_1^\top, \ldots, \alpha_K^\top)^\top$ and $\kappa = (\kappa_1^\top, \ldots, \kappa_K^\top)^\top$ are $n \times 1$ parameter vectors.

We say ζ is distributed as $\mathrm{GCM}(\mu, L, \alpha, \kappa; \psi)$. A conditional GCM density up to a normalizing constant is obtained analogous to (A5) and denoted as $\mathrm{GCM_c}(\mu^*, A_1, \alpha, \kappa; \psi)$. We use these distributions for building the hierarchical models in the following sections. In general, we may be unable to sample directly from either the conditional CM or the conditional GCM distributions except for some familiar exceptions (e.g., the conditional distribution of Gaussian is indeed Gaussian). However, it is possible to consider an augmented model with a particular structure that yields a posterior distribution in the GCM family that is easy to sample from.

B. Examples of Cholesky algorithms

We show an example of a rank-1 update on a $n \times n$ positive definite matrix A with a vector v. We find the Cholesky factor of the matrix $\alpha A + \beta vv^\top$ for $\alpha = \beta = 1$ separately using our proposed function as well as the base R function `chol` and check numerical equality.

```
R> set.seed(1729)
R> tol <- 1E-12
R> n <- 10
R> A <- matrix(rnorm(n^2), n, n)
R> A <- crossprod(A)
R> cholA <- chol(A)
R> v <- 1:n
R> APlusvvT <- A + tcrossprod(v)
R> cholA1 <- t(chol(APlusvvT))
R> cholA2 <- cholUpdateRankOne(cholA, v, alpha = 1, beta = 1, lower = F)
R> print(max(abs(cholA1 - cholA2)) < tol)

TRUE
```

Thus, we test the correctness of the `cholUpdateRankOne()` function. We show results of similar tests for `cholUpdateDel()` and `cholUpdateDelBlock()` in the example code below. In case of `cholUpdateRankOne()`, `ind = 2` corresponds to deletion of the second row/column.

```
R> ind <- 2
R> A1 <- A[-ind, -ind]
R> cholA3 <- t(chol(A1))
R> cholA4 <- cholUpdateDel(cholA, del.index = ind, lower = F)
R> print(max(abs(cholA3 - cholA4)) < tol)
```

```
TRUE
```

In case of `cholUpdateDelBlock`, the indices to be removed is specified by the start index `del.start` and the end index `del.end`. For example, `del.start = 2` and `del.end = 6` corresponds to removal of rows and columns pertaining to the indices $\{2, 3, 4, 5, 6\}$.

```
R> start_ind <- 2
R> end_ind <- 6
R> del_ind <- c(start_ind:end_ind)
R> A2 <- A[-del_ind, -del_ind]
R> cholA5 <- t(chol(A2))
R> cholA6 <- cholUpdateDelBlock(cholA, del.start = start_ind,
+     del.end = end_ind, lower = F)
R> print(max(abs(cholA5 - cholA6)) < tol)
```

```
TRUE
```

Affiliation:

Soumyakanti Pan
Department of Biostatistics
University of California Los Angeles
650 Charles E. Young Drive South
Los Angeles, CA 90095-1772
E-mail: span18@ucla.edu
URL: https://span-18.github.io/

Sudipto Banerjee
Professor of Biostatistics *and*
Professor of Statistics & Data Science
University of California Los Angeles
650 Charles E. Young Drive South
Los Angeles, CA 90095-1772
E-mail: sudipto@ucla.edu
URL: http://sudipto.bol.ucla.edu/