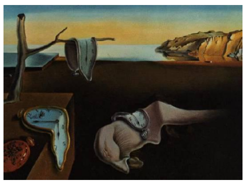


Journal of Statistical Software

December 2018, Volume 87, Issue 13. *doi: 10.18637/jss.v087.i13*

gcKrig: An R Package for the Analysis of Geostatistical Count Data Using Gaussian Copulas

Zifei Han
Vertex Pharmaceuticals

Victor De Oliveira
The University of Texas at San Antonio

Abstract

This work describes the R package **gcKrig** for the analysis of geostatistical count data using Gaussian copulas. The package performs likelihood-based inference and spatial prediction using Gaussian copula models with discrete marginals. Two different classes of methods are implemented to evaluate/approximate the likelihood and the predictive distribution. The package implements the computationally intensive tasks in C++ using an R/C++ interface, and has parallel computing capabilities to predict the response at multiple locations simultaneously. In addition, **gcKrig** also provides functions to simulate and visualize geostatistical count data, and to compute the correlation function of the counts. It is designed to allow a flexible specification of both the marginals and the spatial correlation function. The principal features of the package are illustrated by three data examples from ecology, agronomy and petrology, and a comparison between **gcKrig** and two other R packages.

Keywords: Gaussian copulas, geostatistics, kriging, likelihood inference, parallel computing.

1. Introduction

Spatial data arise in numerous scientific disciplines. Following the classification of spatial data in Cressie (1993), we consider here geostatistical data with discrete responses, specifically geostatistical count data. This type of data appears in many applications such as species counts, disease counts, and mineral indicators. The number and variety of models for geostatistical count data are quite limited, when compared to models available for geostatistical continuous data, and hence the statistical software options for the former are also limited. The two main classes of models for the analysis of geostatistical count data are the hierarchical models proposed by Diggle, Tawn, and Moyeed (1998), and the Gaussian copula models proposed by Madsen (2009), Kazianka and Pilz (2010), and Kazianka (2013a). In this work we consider Gaussian copula models.

Likelihood-based inference for Gaussian copula models is computationally challenging, because the likelihood function can only be expressed as a high dimensional multivariate normal integral. The initial strategies to circumvent likelihood computations were to base inference on surrogate likelihoods that are less computationally demanding. Prime examples of these for the analysis of geostatistical count data are the pairwise likelihood (PL) method proposed in Kazianka and Pilz (2010) and Bai, Kang, and Song (2014), and the generalized quantile transform (GQT) method proposed in Kazianka (2013a). The former uses bivariate copulas only, which is a particular case of the composite likelihood method (see Varin, Reid, and Firth 2011 for a review), while the latter uses a continuous approximation based on copula densities. As for software implementation, Kazianka (2013b) implemented both methods in the MATLAB (The MathWorks Inc. 2017) toolbox **spatialCopula**, and Bai *et al.* (2014) implemented the pairwise likelihood method in the R (R Core Team 2018) package **geoCopula** (Kang, Bai, and Song 2014) for the analysis of spatial-clustered data[1].

A different approach consists of evaluating the likelihood via Monte Carlo simulation, which is called the simulated likelihood method. An example of this approach is the use of a sequential importance sampling algorithm based on a variant of the so-called Geweke-Hajivassiliou-Keane (GHK) simulator (Geweke 1991; Hajivassiliou, McFadden, and Ruud 1996; Keane 1994). Masarotto and Varin (2012) implemented this approach in the R package **gcmr** (Masarotto and Varin 2017, 2018) to analyze longitudinal, time series and spatial data using Gaussian copulas; see also Han and De Oliveira (2019). Another example of this approach is to use a quasi-Monte Carlo algorithm to evaluate the likelihood, which is implemented in the R package **mvtnorm** (Genz and Bretz 2009; Genz, Bretz, Miwa, Mi, and Hothorn 2018). Nikoloulopoulos (2013, 2016) used the latter to make inference about Gaussian copula models with discrete marginals.

The aforementioned packages allow to carry out some of the main tasks of scientific interest for the analysis of geostatistical count data, but not others. In some applications, predictive inference about spatially varying counts at unobserved locations is a task of equal or greater importance than inference about the model parameters. But except for **spatialCopula**, these packages carry out inference only about model parameters. Other tasks of scientific interest are the computation of the correlation function of the process of observed counts, and the determination of its restrictions for a given set of marginal distributions. Closed-form expressions for these are usually not available for Gaussian copula models, and the required numerical computations are not implemented in the aforementioned packages.

The R package **gcKrig** (Han 2018) carries out most of the main tasks of scientific interest for the analysis of geostatistical count data based on Gaussian copula models and it is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=gcKrig`. It offers tools that range from simulation and visualization to estimation and prediction. First, **gcKrig** can compute approximate maximum likelihood estimates and confidence intervals for the model parameters, using either the GHK simulator or the less computationally intensive option based on the GQT method. The latter is appealing for the analysis of moderately large datasets. Second, the package can compute plug-in predictors and prediction intervals at a given set of locations, using either the GHK simulator or the GQT method for the computation of predictive distributions. The prediction tasks are based on parallel computation capabilities, an appealing feature since prediction is usually sought for a

[1]At the time of writing **geoCopula** is not available from the Comprehensive R Archive Network (CRAN), but can be obtained from `http://www-personal.umich.edu/~jiankang/software/GeoCopula.html`.

large number of locations. Third, **gcKrig** can compute the correlation function of the process of counts and the upper bound of this correlation for a given set of marginal distributions (the so-called Fréchet-Hoeffding upper bound), using either a series expansion or a Monte Carlo method. Finally, the package also includes functions for simulation and visualization of spatial data from Gaussian copula models.

This article provides a detailed description of **gcKrig**, illustrates its use with three real-world geostatistical count datasets, and ends with a comparison between it and two other R packages, **mvtnorm** and **gcmr**, which can also perform some of the tasks implemented in **gcKrig**.

2. The Gaussian copula random field

Let $\{Y(\mathbf{s}) : \mathbf{s} \in D\}$, $D \subset \mathbb{R}^2$, be a random field taking values on $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$, and $\{F_{\mathbf{s}}(\cdot\, ; \boldsymbol{\psi}) : \mathbf{s} \in D\}$ be a family of marginal cumulative distribution functions (cdfs) with support contained in $\mathbb{N}_0$ and corresponding probability mass functions (pmfs) $f_{\mathbf{s}}(\cdot\, ; \boldsymbol{\psi})$, where $\boldsymbol{\psi}$ are *marginal parameters*. We assume that $F_{\mathbf{s}}(\cdot\, ; \boldsymbol{\psi})$ is parameterized in terms of its expected value $\mathsf{E}(Y(\mathbf{s}))$, which is given by

$$\mathsf{E}(Y(\mathbf{s})) = t(\mathbf{s}) g^{-1}(\boldsymbol{\beta}^\top \mathbf{f}(\mathbf{s})), \tag{1}$$

where $g^{-1}(\cdot)$ is the inverse of a suitable link function, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$ are regression parameters, $\mathbf{f}(\mathbf{s}) = (f_1(\mathbf{s}), \ldots, f_p(\mathbf{s}))^\top$ are known location-dependent covariates, and $t(\mathbf{s})$ is a known "sampling effort". The cdf $F_{\mathbf{s}}(\cdot\, ; \boldsymbol{\psi})$ can also depend on other marginal parameters, for example on a dispersion parameter $\sigma^2 > 0$, assumed to be constant in D. Typical examples are the negative binomial and zero-inflated Poisson distributions, for which it holds that

$$\mathsf{VAR}(Y(\mathbf{s})) = \mathsf{E}(Y(\mathbf{s}))(1 + \sigma^2 \mathsf{E}(Y(\mathbf{s}))); \tag{2}$$

see Section 3.2 for the marginal families of distributions implemented in **gcKrig**.

The Gaussian copula random field $Y(\cdot)$ is defined by the property that for every $n \in \mathbb{N}$ and $\mathbf{s}_1, \ldots, \mathbf{s}_n \in D$, the joint cdf of $(Y(\mathbf{s}_1), \ldots, Y(\mathbf{s}_n))$ is given by

$$\mathsf{P}(Y(\mathbf{s}_1) \le y_1, \ldots, Y(\mathbf{s}_n) \le y_n) = \Phi_n(\Phi^{-1}(F_{\mathbf{s}_1}(y_1; \boldsymbol{\psi})), \ldots, \Phi^{-1}(F_{\mathbf{s}_n}(y_n; \boldsymbol{\psi})); \Psi_{\boldsymbol{\vartheta}}), \tag{3}$$

where $\Phi(\cdot)$ is the cdf of the standard normal distribution and $\Phi_n(\cdot\, ; \Psi_{\boldsymbol{\vartheta}})$ is the cdf of the $\mathrm{N}_n(\mathbf{0}, \Psi_{\boldsymbol{\vartheta}})$ distribution. The (i, j)th entry of the $n \times n$ matrix $\Psi_{\boldsymbol{\vartheta}}$ is assumed to be

$$K_{\boldsymbol{\vartheta}}(d_{ij}) = (1 - \tau^2)\bar{K}_{\boldsymbol{\theta}}(d_{ij}) + \tau^2 \mathbf{1}\{d_{ij} = 0\}, \tag{4}$$

where $\bar{K}_{\boldsymbol{\theta}}(d_{ij}) \ge 0$ is an isotropic correlation function in $\mathbb{R}^2$ that is continuous everywhere, $d_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\|$ is the Euclidean distance, $\boldsymbol{\vartheta} = (\boldsymbol{\theta}^\top, \tau^2)$ are correlation parameters, with $\boldsymbol{\theta}$ the parameter(s) appearing in $\bar{K}_{\boldsymbol{\theta}}(\cdot)$, $\tau^2 \in [0, 1]$ the so-called *nugget effect* (so $\Psi_{\boldsymbol{\vartheta}}$ is a correlation matrix), and $\mathbf{1}\{A\}$ is the indicator function of A; see Section 3.2 for the correlation functions implemented in **gcKrig**.

An alternative formulation of the above model consists of viewing it as a transformed Gaussian random field (De Oliveira 2003). Consider the latent Gaussian random field $\{Z(\mathbf{s}) : \mathbf{s} \in D\}$

with mean 0, variance 1 and correlation function $K_{\boldsymbol{\vartheta}}(d_{ij})$. Then the Gaussian copula random field $Y(\cdot)$ can be written as

$$Y(\mathbf{s}) = F_{\mathbf{s}}^{-1}(\Phi(Z(\mathbf{s}))\ ;\ \boldsymbol{\psi}), \qquad \mathbf{s} \in D, \tag{5}$$

where $F_{\mathbf{s}}^{-1}(\cdot\ ;\ \boldsymbol{\psi})$ is the quantile function defined as

$$F_{\mathbf{s}}^{-1}(u\ ;\ \boldsymbol{\psi}) = \inf\{x \in \mathbb{R} : F_{\mathbf{s}}(x\ ;\ \boldsymbol{\psi}) \geq u\}, \qquad u \in (0,1). \tag{6}$$

Unlike the hierarchical models proposed by Diggle *et al.* (1998), Gaussian copula models allow the separate modeling of the marginal and dependence structures of the data. The marginals (and hence the mean function) of the random field are modeled explicitly by the family of cdfs $\{F_{\mathbf{s}}(\cdot\ ;\boldsymbol{\psi}) : \mathbf{s} \in D\}$, while the dependence structure is modeled through the family of correlation functions $K_{\boldsymbol{\vartheta}}(\cdot)$, albeit not in an explicit way; see Section 3.5 for the correlation function of $Y(\cdot)$.

Remark. Genest and Nešlehová (2007) caution modelers about some potential limitations of the use of copula models to describe multivariate discrete data, since many of the properties of these models that hold when the marginals are continuous, do not hold when the marginals are discrete. One of these is the uniqueness of the copula that couples the marginal cdfs to produce a given joint cdf; this, in general, may pose an identifiability problem. This problem does *not* arise in model (3) though, since this model restricts consideration to Gaussian copulas, and for any $u_1, \ldots, u_n$ the copula $\Phi_n(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_n); \Psi_{\boldsymbol{\vartheta}})$ is a one-to-one function of $\Psi_{\boldsymbol{\vartheta}}$.

2.1. Likelihood computation

The package **gcKrig** implements two methods for likelihood computation, one approximates the likelihood function via Monte Carlo simulation, and the other replaces the likelihood function with a surrogate likelihood. Let $\boldsymbol{\eta} = (\boldsymbol{\psi}, \boldsymbol{\vartheta})$ denote all the parameters in the model. For count responses the likelihood of $\boldsymbol{\eta}$ can be written as the n-dimensional normal integral

$$L(\boldsymbol{\eta}; \boldsymbol{y}) = \int_{\zeta_{\mathbf{s}_1}(y_1-1;\boldsymbol{\psi})}^{\zeta_{\mathbf{s}_1}(y_1;\boldsymbol{\psi})} \cdots \int_{\zeta_{\mathbf{s}_n}(y_n-1;\boldsymbol{\psi})}^{\zeta_{\mathbf{s}_n}(y_n;\boldsymbol{\psi})} \phi_n(z_1, \ldots, z_n; \Psi_{\boldsymbol{\vartheta}})\, dz_1 \ldots dz_n, \tag{7}$$

where $\zeta_{\mathbf{s}}(y; \boldsymbol{\psi}) = \Phi^{-1}(F_{\mathbf{s}}(y; \boldsymbol{\psi}))$ and $\phi_n(\cdot\ ; \Psi_{\boldsymbol{\vartheta}})$ is the probability density function (pdf) of the $\mathrm{N}_n(\mathbf{0}, \Psi_{\boldsymbol{\vartheta}})$ distribution. To approximate the integral in Equation 7, **gcKrig** implements a variant of the popular sequential importance sampling algorithm proposed by Geweke (1991), Hajivassiliou *et al.* (1996) and Keane (1994), known as the GHK simulator. The integral in Equation 7 is approximated by importance sampling, using the importance sampling density with support $(\zeta_{\mathbf{s}_1}(y_1 - 1), \zeta_{\mathbf{s}_1}(y_1)) \times \cdots \times (\zeta_{\mathbf{s}_n}(y_n - 1), \zeta_{\mathbf{s}_n}(y_n))$ given by

$$g_{\boldsymbol{\vartheta}}(\boldsymbol{z}) = \prod_{i=1}^{n} p_{\boldsymbol{\vartheta}}(z_i \mid z_{i-1}, \ldots, z_1, y_i), \tag{8}$$

where $p_{\boldsymbol{\vartheta}}(z_i \mid z_{i-1}, \ldots, z_1, y_i)$ is the conditional density of $Z(\mathbf{s}_i)$ given $Z(\mathbf{s}_{i-1}), \ldots, Z(\mathbf{s}_1)$, and $Y(\mathbf{s}_i) = y_i$, and $Z(\cdot)$ is the Gaussian random field defined just before Equation 5. Since $Z(\mathbf{s}_i) \mid Z(\mathbf{s}_{i-1}), \ldots, Z(\mathbf{s}_1) \sim \mathrm{N}(m_i, v_i^2)$, with $m_i = \mathsf{E}_{\boldsymbol{\vartheta}}(Z(\mathbf{s}_i) \mid Z(\mathbf{s}_{i-1}), \ldots, Z(\mathbf{s}_1)) = m_i(Z(\mathbf{s}_{i-1}), \ldots, Z(\mathbf{s}_1); \boldsymbol{\vartheta})$ and $v_i^2 = \mathrm{var}_{\boldsymbol{\vartheta}}(Z(\mathbf{s}_i) \mid Z(\mathbf{s}_{i-1}), \ldots, Z(\mathbf{s}_1)) = v_i^2(\boldsymbol{\vartheta})$, it follows from

Equation 5 that $p_{\boldsymbol{\vartheta}}(z_i \mid z_{i-1}, \dots, z_1, y_i)$ is the density of the $\mathrm{N}(m_i, v_i^2)$ distribution truncated to the interval $(\zeta_{\mathbf{s}_i}(y_i - 1; \boldsymbol{\psi}), \zeta_{\mathbf{s}_i}(y_i; \boldsymbol{\psi}))$. Then the GHK simulator approximates $L(\boldsymbol{\eta}; \boldsymbol{y})$ by

$$\hat{L}(\boldsymbol{\eta}; \boldsymbol{y}) = \frac{1}{M} \sum_{k=1}^{M} \left\{ \prod_{i=1}^{n} \left[\Phi\left(\frac{\zeta_{\mathbf{s}_i}(y_i; \boldsymbol{\psi}) - m_{ki}(\boldsymbol{\vartheta})}{v_{ki}(\boldsymbol{\vartheta})} \right) - \Phi\left(\frac{\zeta_{\mathbf{s}_i}(y_i - 1; \boldsymbol{\psi}) - m_{ki}(\boldsymbol{\vartheta})}{v_{ki}(\boldsymbol{\vartheta})} \right) \right] \right\}, \tag{9}$$

where $\boldsymbol{Z}^{(1)}, \dots, \boldsymbol{Z}^{(M)}$ are i.i.d. draws from $g_{\boldsymbol{\vartheta}}(\boldsymbol{z})$, with $\boldsymbol{Z}^{(k)} = (Z_1^{(k)}, \dots, Z_n^{(k)})$, $m_{ki}(\boldsymbol{\vartheta}) = m_i(Z_{i-1}^{(k)}, \dots, Z_1^{(k)}; \boldsymbol{\vartheta})$, and $v_{ki}^2(\boldsymbol{\vartheta}) = v_i^2(\boldsymbol{\vartheta})$. The simulation of each $\boldsymbol{Z}^{(k)}$ is done sequentially, $Z_1^{(k)}, Z_2^{(k)}, \dots, Z_n^{(k)}$, using the standard algorithm for simulating from truncated normal distributions. This method is also implemented in the package **gcmr** (Masarotto and Varin 2017, 2018). The GHK simulator requires an ordering of the sampling locations, which for geostatistical data is arbitrary. Although empirical experiments suggest that parameter estimates are not sensitive to different orderings, **gcKrig** provides the option to use the ordering obtained by the algorithm in Gibson, Glasbey, and Elston (1994), which orders the sampling locations based on the impact that their corresponding observations have on the likelihood (integral); see Genz and Bretz (2009) and Ridgway (2016) for details.

Many other approaches have been proposed for the approximation of n-dimensional normal integrals, like those in Equation 7. One of these is the quasi-Monte Carlo approximation implemented in the R package **mvtnorm** (Genz and Bretz 2009; Genz *et al.* 2018), and used for Gaussian copula estimation by Nikoloulopoulos (2013, 2016). Another is a recent approach that uses the minimax tilting algorithm proposed by Botev (2017); see also Genz and Bretz (2009) for a review of classical approximations. A simulation experiment carried out by Han and De Oliveira (2019), comparing the GHK simulator, quasi-Monte Carlo approximation and a third method, showed that the GHK simulator provides the best balance between statistical efficiency and computational efficiency.

A different strategy consists of avoiding the calculation of the likelihood in Equation 7 altogether, and instead base inference on a surrogate likelihood. The **gcKrig** package also implements the so-called generalized quantile transformation (GQT) method proposed in Kazianka and Pilz (2010) and Kazianka (2013a,b), which is (partially) implemented in the MATLAB toolbox **spatialCopula**. In this strategy the likelihood is replaced by the surrogate likelihood

$$|\Psi_{\boldsymbol{\vartheta}}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \Phi^{-1}(\boldsymbol{u})^\top \left(\Psi_{\boldsymbol{\vartheta}}^{-1} - \mathbf{I}_n \right) \Phi^{-1}(\boldsymbol{u}) \right) \prod_{i=1}^{n} f_{\mathbf{s}_i}(y_i; \boldsymbol{\psi}), \tag{10}$$

where $\Phi^{-1}(\boldsymbol{u}) = (\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_n))^\top$ and $u_i = (F_{\mathbf{s}_i}(y_i - 1; \boldsymbol{\psi}) + F_{\mathbf{s}_i}(y_i; \boldsymbol{\psi}))/2$.

The **gcKrig** package approximates the maximum likelihood estimates by optimizing the right-hand-side of either Equation 9 or 10, using the box-constrained quasi-Newton algorithm implemented in the R function `optim()`. The asymptotic standard errors are computed in the classical way, by calculating the inverse of the (approximate) log-likelihood Hessian matrix evaluated at the estimates. It should be noted that when the "discreteness" in the data is extreme, e.g., when analyzing binary data, the GQT approximation becomes highly inaccurate (Kazianka 2013a), so parameter estimates and their estimated variances are not reliable. In general we recommend using the GHK simulator as the preferred option, if the size of the data is not too large and the analysis not too time consuming, and leaving the GQT method for moderately larger datasets.

2.2. Predictive inference

Let $\hat{\boldsymbol{\eta}}$ be the vector of parameter estimates, and $\mathbf{s}_0 \in D$ a location for which $Y(\cdot)$ is not observed. The (plug-in) predictive distribution of $Y(\mathbf{s}_0)$ is defined as

$$\mathsf{P}(Y(\mathbf{s}_0) = y_0 \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}}) = \frac{\mathsf{P}(Y(\mathbf{s}_0) = y_0, Y(\mathbf{s}_1) = y_1, \ldots, Y(\mathbf{s}_n) = y_n; \hat{\boldsymbol{\eta}})}{\mathsf{P}(Y(\mathbf{s}_1) = y_1, \ldots, Y(\mathbf{s}_n) = y_n; \hat{\boldsymbol{\eta}})}, \quad y_0 \in \mathbb{N}_0. \tag{11}$$

Once the predictive distribution is computed for all y_0 in the (practical) support of Equation 11, one can compute different optimal predictors, depending on the loss function that is assumed. Under the squared error loss, the optimal predictor is $\hat{Y}(\mathbf{s}_0) = \mathsf{E}(Y(\mathbf{s}_0) \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}})$. This is guaranteed to be non-negative, but not integer-valued. According to Jeske (1993), the optimal integer-valued predictor is given by

$$\hat{Y}(\mathbf{s}_0) = \lfloor 0.5 + \mathsf{E}(Y(\mathbf{s}_0) \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}}) \rfloor = \left\lfloor 0.5 + \sum_{y_0=0}^{\infty} \mathsf{P}(Y(\mathbf{s}_0) = y_0 \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}}) y_0 \right\rfloor, \tag{12}$$

where $\lfloor \cdot \rfloor$ is the floor function, and the uncertainty measure associated to this predictor is

$$\widehat{\mathsf{VAR}}(Y(\mathbf{s}_0) \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}}) = \mathsf{E}(Y^2(\mathbf{s}_0) \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}}) - \left(\mathsf{E}(Y(\mathbf{s}_0) \mid \boldsymbol{Y}; \hat{\boldsymbol{\eta}})\right)^2. \tag{13}$$

Kazianka and Pilz (2010) and Kazianka (2013a) suggested to approximate the numerator and denominator of the predictive distribution using the GQT method, as in Equation 10. The computational speed of this alternative is fast, but the approximation error can be substantial in the case of low marginal variance and strong spatial dependence (Nikoloulopoulos 2016). A more precise alternative is to compute the numerator and denominator of the predictive distribution in Equation 11 using the GHK simulator. The package **gcKrig** implements both alternatives. Since spatial prediction is usually needed for a large number of locations, **gcKrig** speeds up the computations with the help of parallel computing techniques using the R package **snowfall** (Knaus 2015); see Section 3.4 for details.

2.3. Correlation function of the Gaussian copula random field

Recall that $K_{\boldsymbol{\vartheta}}(d)$ is the correlation function of the latent Gaussian process $Z(\cdot)$ in Equation 5. The correlation function of the Gaussian copula random field $Y(\cdot)$ is not available in closed form for most families of marginals. The package **gcKrig** implements the series expansion discussed in De Oliveira (2013) to approximate the covariance (and correlation) function of $Y(\cdot)$. Specifically

$$\mathsf{COV}\{Y(\mathbf{s}), Y(\mathbf{u})\} = \sum_{k=1}^{\infty} a_k(F_{\mathbf{s}}) a_k(F_{\mathbf{u}}) \frac{\left(K_{\boldsymbol{\vartheta}}(\|\mathbf{s} - \mathbf{u}\|)\right)^k}{k!}, \qquad \mathbf{s}, \mathbf{u} \in D, \tag{14}$$

with

$$a_k(F_{\mathbf{s}}) = \int_{-\infty}^{\infty} F_{\mathbf{s}}^{-1}(\Phi(t); \boldsymbol{\psi}) H_k(t) \phi(t) dt, \qquad k = 1, 2, \ldots \tag{15}$$

where $\phi(t)$ is the pdf of the standard normal distribution and $H_k(t)$ is the (probabilists') Hermite polynomial of degree k ($H_1(t) = t$, $H_2(t) = t^2 - 1$, …). Hence, $\mathsf{COR}\{Y(\mathbf{s}), Y(\mathbf{u})\}$ can be quickly and accurately approximated by truncating the series in Equation 14 and approximating the coefficients in Equation 15 by numerical quadrature (e.g., using the R function `integrate()`). Han and De Oliveira (2016) used this expansion to investigate properties of this correlation function, founding it to be flexible in several regards; see Section 3.5 for computational details.

3. Package functionality

The R package **gcKrig** offers a number of tools for the analysis of geostatistical count data, ranging from simulation and visualization to estimation and prediction, with the two core functions being `mlegc()` (for parameter estimation) and `predgc()` (for spatial prediction).

3.1. Data simulation and visualization

The function `simgc()` simulates geostatistical count data at a given set of m locations from Gaussian copula models with different marginal and correlations structures:

```
simgc(locs, sim.n = 1, marginal, corr, longlat = FALSE)
```

The argument `locs` is a numeric $m \times 2$ matrix or data frame containing the coordinates of the locations, `sim.n` is a positive integer indicating the number of simulated datasets. Also, `marginal` is an argument specified with an object of class '`marginal.gc`' indicating the family of marginals, and `corr` is an argument specified with an object of class '`corr.gc`' indicating the family of correlation functions; see Section 3.2. If `longlat = TRUE`, the great circle distance is used in the correlation function; otherwise the Euclidean distance is used. The function returns a list of class '`simgc`' with the simulated data and the corresponding locations.

To visualize a simulated geostatistical count dataset, an S3 method `plot` is available for objects of class '`simgc`':

```
plot(x, index = 1, plottype = "Text", xlab = "xloc", ylab = "yloc",
  xlim = NULL, ylim = NULL, pch = 20, textcex = 0.8, plotcex = 1,
  angle = 60, col = 4, col.regions = gray(90:0/100), ...)
```

The argument `x` is an object of class '`simgc`' generated from the function `simgc()`, `index` is the index of the simulated dataset, `plottype` is one of the following, `"Text"`, `"Dot"` or `"3D"`, denoting the type of the plot, and `col.regions` is the color vector to be used that represents the magnitude of the observations at the locations. The general strategy for a good visualization is to set the color vector with gradually varying colors. The other arguments have the same meanings as in the standard R function `plot()`. The function can generate three types of plot, allowing the visualization of different aspects of the dataset; see Figure 1 and its legend.

```
R> library("gcKrig")
R> grid <- seq(0.05, 0.95, by = 0.1)
R> xloc <- expand.grid(x = grid, y = grid)[, 1]
R> yloc <- expand.grid(x = grid, y = grid)[, 2]
R> set.seed(12345)
```

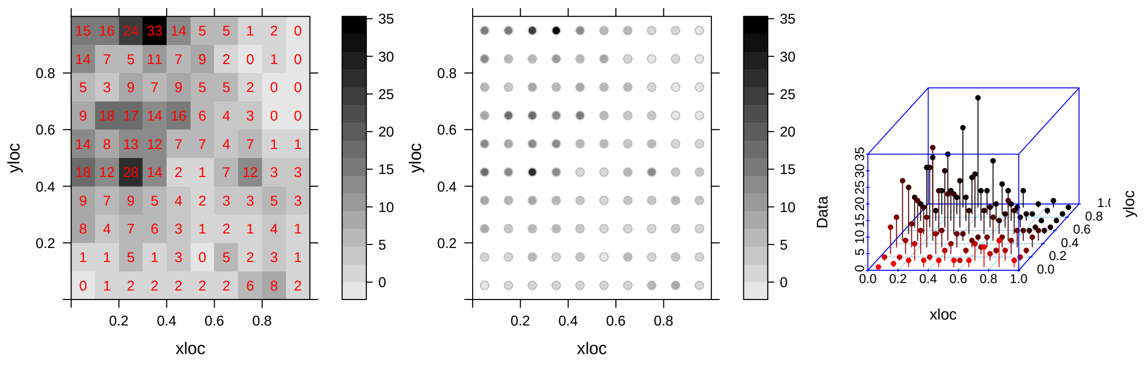


Figure 1: Visualization of a simulated dataset. Left: Gray scale plot with the number of counts at the locations (`plottype = "Text"`). Middle: Gray scale plot with point referenced locations and varying colors to denote magnitude of counts (`plottype = "Dot"`). Right: A 3-D scatter plot of the simulated data (`plottype = "3D"`)
.

```
R> sim1 <- simgc(locs = cbind(xloc,yloc), sim.n = 10,
+    marginal = negbin.gc(mu = 5, od = 1),
+    corr = matern.gc(range = 0.3, kappa = 0.5, nugget = 0.1))
R> plot(sim1, index = 1, plottype = "Text", col = 2)
R> plot(sim1, index = 1, plottype = "Dot", col = 2)
R> plot(sim1, index = 1, plottype = "3D", col = 2)
```

3.2. Classes of marginals and correlations

We describe here how to set the families of marginal and correlation functions needed to specify Gaussian copula models in **gcKrig**. To make the package flexible, the arguments in the **gcKrig**'s functions that specify marginals and correlations are of class '`marginal.gc`' and '`corr.gc`', respectively. These arguments are needed in both the functions `simgc()` and `corrTG()` that are used to simulate geostatistical data and compute correlations between the non-Gaussian variables, as well as in `mlegc()` and `predgc()` that are used to make inference about the model parameters and spatial prediction. For simulation of geostatistical count data or computation of correlations between the count variables, the parameters need to be set as arguments in the corresponding functions. For inference about the model parameters and spatial prediction the parameters are not set, as they are estimated from the data, which is the default setting for discrete marginals and all correlation functions. In this case, the link functions are also specified as arguments. When fitting a correlation model, the nugget effect can be estimated (`nugget = TRUE`) or set to zero (`nugget = FALSE`), with the former being the default. Also, for some correlation functions a shape parameter `kappa` needs to be specified. Table 1 lists all the marginals and correlation functions available in **gcKrig**.

Standard parameterizations are used for most families of marginals, except for the negative binomial and zero-inflated Poisson marginals. These are parameterized in terms of their mean

Name	Distribution/ Correlation
`poisson.gc(link = "log", lambda = NULL)`	Poisson
`negbin.gc(link = "log", mu = NULL, od = NULL)`	negative binomial
`zip.gc(link = "log", mu = NULL, od = NULL)`	zero-inflated Poisson
`binomial.gc(link = "logit", size = NULL, prob = NULL)`	binomial
`gaussian.gc(mean = 0, sd = 1)`	Gaussian
`gm.gc(shape = 1, rate = 1)`	Gamma
`beta.gc(shape1 = 1, shape2 = 1)`	Beta
`weibull.gc(shape = 1, scale = 1)`	Weibull
`matern.gc(range = NULL, kappa = 0.5, nugget = TRUE)`	Matérn
`powerexp.gc(range = NULL, kappa = 1, nugget = TRUE)`	power exponential
`spherical.gc(range = NULL, nugget = TRUE)`	spherical

Table 1: Marginals of class ‘`marginal.gc`’ and correlation functions of class ‘`corr.gc`’.

$\mu(\mathbf{s})$ and overdispersion σ^2, with respective pmfs

$$f_{\mathbf{s}}^{\mathrm{NB}}(y;\boldsymbol{\psi}) = \frac{1}{B(y, 1/\sigma^2)} \left(\frac{1}{1+\mu(\mathbf{s})\sigma^2}\right)^{1/\sigma^2} \left(\frac{\mu(\mathbf{s})\sigma^2}{1+\mu(\mathbf{s})\sigma^2}\right)^{y}, \quad y = 0, 1, 2, \ldots \tag{16}$$

$$f_{\mathbf{s}}^{\mathrm{ZIP}}(y;\boldsymbol{\psi}) = \begin{cases} \frac{\sigma^2}{1+\sigma^2} + \frac{1}{1+\sigma^2} \exp\left(-(1+\sigma^2)\mu(\mathbf{s})\right) & \text{if } y = 0, \\ \frac{1}{1+\sigma^2} \frac{\exp\left(-\left(1+\sigma^2\right)\mu(\mathbf{s})\right)\left(1+\sigma^2\right)^y (\mu(\mathbf{s}))^y}{y!} & \text{if } y = 1, 2, \ldots \end{cases} \tag{17}$$

in which case the mean-variance relation in Equation 2 holds; see Cameron and Trivedi (2013) and Han and De Oliveira (2016).

Three families of isotropic correlation functions are implemented in **gcKrig**: the *Matérn* family given by

$$\bar{K}(d) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)} \left(\frac{d}{\phi}\right)^{\kappa} \mathcal{K}_{\kappa}\left(\frac{d}{\phi}\right), \qquad d \geq 0, \quad \phi > 0, \quad \kappa > 0, \tag{18}$$

where $\Gamma(\cdot)$ is the gamma function and $\mathcal{K}_\kappa(\cdot)$ is the modified Bessel function of the second kind and order κ; the *power exponential* family given by

$$\bar{K}(d) = \exp\Big\{ - \Big(\frac{d}{\phi}\Big)^{\kappa} \Big\}, \qquad d \geq 0, \quad \phi > 0, \quad \kappa \in (0, 2], \tag{19}$$

and the *spherical* family given by

$$\bar{K}(d) = \left(1 - \frac{3}{2}\left(\frac{d}{\phi}\right) + \frac{1}{2}\left(\frac{d}{\phi}\right)^3\right) \mathbf{1}_{[0,\phi]}(d), \qquad d \geq 0, \quad \phi > 0. \tag{20}$$

For these families ϕ is called a *range* parameter, which controls the rate of correlation decay, and κ is called a *shape* parameter, which controls the smoothness of the random field.

3.3. Estimation

The core function for parameter estimation is `mlegc()`, which provides either maximum simulated likelihood estimates computed via the GHK simulator (Masarotto and Varin 2012) or maximum surrogate likelihood estimates computed via the generalized quantile transform (Kazianka and Pilz 2010):

```
mlegc(y, x = NULL, locs, marginal, corr, effort = 1, longlat = FALSE,
  distscale = 1, method = "GHK", corrpar0 = NULL,
  ghkoptions = list(nrep = c(100, 1000), reorder = FALSE, seed = 12345))
```

The argument `y` is a vector containing the response counts, `x` is a matrix or data frame containing the covariates (`x = NULL` when no covariates are available), and `locs` is a matrix or data frame containing the coordinates of the sampling locations. By default these are assumed Cartesian coordinates. When longitude and latitude coordinates are used, then `longlat = TRUE` should be set. The argument `marginal` is a function of class '`marginal.gc`' that specifies the marginal distributions of the counts, and `corr` is a function of class '`corr.gc`' that specifies the correlation function defining the Gaussian copula; the current version of **gcKrig** includes the four families of discrete marginal distributions and the three families of isotropic correlation functions listed in Table 1. The argument `effort` is a vector containing the sampling efforts associated with the sampling locations. For instance, in the case of binomial marginals these efforts are the "total number of trials"; they are all set to 1 by default. The argument `distscale` is an optional scalar factor to be multiplied by the elements of the distance matrix. For example, if the original distance is measured in kilometers, we can convert this into meters by setting `distscale = 1000`.

The package **gcKrig** includes two fitting methods. If `method = "GQT"` is set, the likelihood is replaced by the generalized quantile transform; if `method = "GHK"` is set, the likelihood is approximated by the GHK simulator. In the latter case the arguments in the list `ghkoptions` need to be specified: `nrep` sets the Monte Carlo size, `seed` sets the seed of the pseudo-random number generator, and `reorder` controls whether or not an ordering of the locations following the algorithm in Gibson *et al.* (1994) is computed before each likelihood evaluation. As in the R package **gcmr**, the argument `nrep` can be a vector with increasing positive integers, in which case the model is fitted several times with different Monte Carlo sizes.

To optimize the likelihood approximation, **gcKrig** uses the quasi-Newton algorithm with box constraints implemented in the R function `optim()`, with `method = "L-BFGS-B"`. The starting values of the marginal parameters are set as the estimates obtained by treating the observations as if they were independent. The starting values of the covariance parameters can be set by the argument `corrpar0`. By default the starting value of the range parameter is set at half the median of the elements of the distance matrix, and the starting value of the nugget parameter is set at 0.2. When the model is fitted several times with different Monte Carlo sizes, the starting values for each fit are set at the parameter estimates from the previous fit. Once the likelihood function evaluated at the maximum likelihood estimates is available, it is possible to conduct model selection using information criteria such as AIC, BIC or AIC_c; these are computed by **gcKrig** and illustrated in Section 4.1.

The function `mlegc()` outputs a variety of objects of class '`mlegc`'. The following S3 methods are available for this class of objects: `summary()` and `print()`, which extract the basic summaries, following the format in the R package **gcmr** (Masarotto and Varin 2017, 2018); `plot()` generates contour plots and 3-D scatter plots of the original data and the fitted mean response; `vcov()` displays the covariances and correlations between parameter estimates, computed from the inverse of the observed Fisher information matrix.

The package **gcKrig** computes both Wald-type and likelihood-based confidence intervals for the parameters, where the latter are computed by evaluating the profile likelihood and inverting a likelihood ratio test; see Meeker and Escobar (1995) for details. The likelihood-based confidence intervals are implemented by the S3 method `profile` for '`mlegc`' objects. The endpoints of the profile likelihood confidence intervals are computed, not by evaluating the profile likelihood on a selected grid of parameter values, as Masarotto and Varin (2017) do, but by using two runs of the Newton-Raphson algorithm, each with a different starting point (see below). The function call is as follows

```
profile(fitted, par.index, alpha = 0.05, start.point = NULL, method = "GQT",
  nrep = 1000, seed = 12345, ...)
```

where `fitted` is the fitted model of class '`mlegc`' and `par.index` is the index of the parameter to be profiled. For example, when `par.index = 1`, the confidence interval of the intercept parameter will be calculated. The argument `alpha` is a scalar between 0 and 1 denoting the confidence level, and `start.point` is a numeric vector of length 2 indicating the starting points of the Newton-Raphson iteration; by default these points are the endpoints of the Wald-type confidence interval. The argument `method` specifies the likelihood evaluation method with choices `"GQT"` and `"GHK"`. If `method = "GHK"`, the Monte Carlo size and the seed of the pseudo-random number generator are specified by `nrep` and `seed`, respectively.

3.4. Prediction

The core function for spatial prediction is `predgc()`, which approximates the predictive distribution in Equation 11 and computes optimal predictors at a set of prediction locations. The prediction task can become computationally intensive since prediction is usually sought for a large number of locations on a grid covering the region of interest. Noting that the denominator in Equation 11 is the same for all prediction locations and the numerators can be computed independently of each other, **gcKrig** assigns the computation of the latter to various cores of the computer using parallel computing techniques. Although by default R will only use one processor regardless of the number of available cores, **gcKrig** takes advantage of the parallel computing techniques from the R package **snowfall** (Knaus 2015). This is carried out by the function `predgc()`:

```
predgc(obs.y, obs.x = NULL, obs.locs, pred.x = NULL, pred.locs,
  longlat = FALSE, distscale = 1, marginal, corr, obs.effort = 1,
  pred.effort = 1, method = "GHK", estpar = NULL, corrpar0 = NULL,
  pred.interval = NULL, parallel = FALSE,
  ghkoptions = list(nrep = c(100,1000), reorder = FALSE, seed = 12345),
  paralleloptions = list(n.cores = 2, cluster.type = "SOCK"))
```

where many of its arguments are the same as those in `mlegc()`. The argument `obs.y` is the vector of observed counts, `obs.x` and `pred.x` are the matrices or data frames of the covariates at the sampling and prediction locations, respectively, and `obs.locs` and `pred.locs` are the matrices or data frames containing the coordinates of the sampling and prediction locations. The arguments `obs.effort` and `pred.effort` are the respective efforts at the sampling and predictive locations. The argument `estpar` is the vector of parameters needed to carry out plug-in prediction. It is set to `NULL` by default, which means the function will call `mlegc` to fit the observed counts first, and then these parameter estimates are used in the plug-in prediction. It is also possible to specify `estpar` as a numeric vector with some preliminary estimates, so the model will not be re-fitted.

The function `predgc()` can also compute prediction intervals, which can be of two types. One of them is the "equal-tail" prediction interval that divides equally the complement of the confidence level to both tails of the predictive distribution in Equation 11. The other is the "highest probability mass" prediction interval that includes in the interval all the values in the support of $Y(\mathbf{s}_0)$ with the highest values in Equation 11. In general, the latter type of prediction interval is recommended, since predictive distributions are often highly asymmetric. When prediction intervals are needed one sets the argument `pred.interval` equal to a number between 0 and 1 representing the confidence level, in which case both types of prediction intervals are computed. By default `predgc()` does not calculate prediction intervals.

When the argument `parallel = FALSE` is set, a serial version of the function will be called, while when `parallel = TRUE` the function calls for parallel computation. The argument `parallel.options` is a list with elements `n.cores` and `cluster.type` that set the number of cores and type of cache for parallel computing, respectively. Four cache types are available: `"SOCK"`, `"MPI"`, `"PVM"` or `"NWS"`, with `"SOCK"` being the default. More details can be found in the manual of the **snowfall** package (Knaus 2015).

The output of the function `predgc()` is a list of objects with class '`predgc`'. The S3 method `summary()` for '`predgc`' objects extracts key summary information, including prediction locations, the mean of the predictive distribution, predicted discrete response, estimated prediction variance, and two types of the prediction intervals, if `pred.interval` is provided. The S3 method `plot` for '`predgc`' objects works in a similar way as the previously described method for '`simgc`' objects, which can generate five plots. One of them is a 3-D scatter plot with both observed and predicted counts. The other four are contour plots displaying the observed number of counts, the estimated means, the predicted responses and estimated prediction variances. The usage of `predgc()` is illustrated with an example in Section 4.

3.5. Computation of correlations

For any two locations the function `corrTG()` computes the correlation between $Y(\mathbf{s})$ and $Y(\mathbf{u})$, either by using the series expansion in Equation 14 or a Monte Carlo method:

```
corrTG(marg1, marg2, corrGauss = 0.5, method = "integral", nrep = 1000)
```

The arguments `marg1` and `marg2` are the marginal distributions of $Y(\mathbf{s})$ and $Y(\mathbf{u})$ from the class '`marginal.gc`', and `corrGauss` is the correlation of the latent Gaussian random field $Z(\cdot)$. If `method = "integral"`, the covariance is computed by using the series expansion in Equation 14. The series is approximated by truncating Equation 14 up to a term so that the magnitude of the next term is smaller than the square root of the smallest positive floating-point number (`sqrt(.Machine$double.eps)`). The coefficients in Equation 15 are computed by the R function `integrate()`. If `method = "mc"`, a Monte Carlo method is used, which consists of two steps. First, a random sample of size `nrep` is generated from the bivariate Gaussian distribution with means $(0, 0)$, variances $(1, 1)$ and correlation in Equation 4 using the function `mvrnorm()` in the R package **MASS** (Venables and Ripley 2002). Second, the transformation in Equation 5 is applied to the bivariate Gaussian draws and the desired correlation is approximated by the sample correlation of the transformed draws. A similar approach was carried out by Demirtas and Hedeker (2011) to compute the Fréchet-Hoeffding upper and lower bounds.

3.6. Computation of Fréchet-Hoeffding upper bounds

The possible correlations between two non-Gaussian random variables are, in general, a subset of $[-1,1]$. The maximum and minimum correlations that are attainable for a given pair of marginal distributions are called the Fréchet-Hoeffding bounds (Nelsen 2006). An important property of Gaussian copula models is that the correlation between two random variables following these models always attains the Fréchet-Hoeffding upper bound, when $\sup\{K_{\boldsymbol{\vartheta}}(d) : d > 0\} = 1$ (Nelsen 2006; Grigoriu 2007). The Fréchet-Hoeffding upper bound can be computed for any pair of marginals with the function `corrTG()`, by setting the argument `corrGauss = 1`.

When both marginal distributions are discrete, there is an alternative expression for the Fréchet-Hoeffding upper bound, proposed by Nelsen (1987), that may be faster to compute in some cases. It is given by

$$\frac{\sum_{(x,y)\in S}\left(1 - F_{\mathbf{u}}(y;\boldsymbol{\psi})\right) + \sum_{(x,y)\in T}\left(1 - F_{\mathbf{s}}(x;\boldsymbol{\psi})\right) \; - \; \mu(\mathbf{s})\mu(\mathbf{u})}{\left(\mathsf{VAR}(Y(\mathbf{s}))\mathsf{VAR}(Y(\mathbf{u}))\right)^{\frac{1}{2}}}, \tag{21}$$

where

$$S = \{(x,y) \in \mathbb{N}_0^2 : F_{\mathbf{s}}(x\;;\;\boldsymbol{\psi}) \le F_{\mathbf{u}}(y;\boldsymbol{\psi})\} \quad , \quad T = S^c. \tag{22}$$

The computation of Equation 21 is faster than the computation of Equation 14 when both marginal means are small. This alternative expression for the upper bound is implemented in the function `FHUBdiscrete()`, where the summation over a two-dimensional grid is coded in C++:

```
FHUBdiscrete(marg1, marg2, mu1, mu2, od1 = 0, od2 = 0, binomial.size1 = 1,
  binomial.size2 = 1)
```

The arguments `marg1` and `marg2` are the names of the possible discrete marginals: `"poisson"`, `"zip"`, `"nb"` and `"binomial"` (for Poisson, zero-inflated Poisson, negative binomial and binomial marginals, respectively). The argument `mu1` is the mean of the first marginal, `od1` is the overdispersion parameter of the first marginal, when this is negative binomial or zero-inflated Poisson, and `binomial.size1` is the number of trials, when this is binomial. The other arguments have the same interpretation, but for the second marginal.

Below are two examples. The first example, using `FHUBdiscrete()`, computes the Fréchet-Hoeffding upper bounds for the correlations between the negative binomial random variable with mean 10 and overdispersion 0.2, and negative binomial random variables with means varying from 0.01 to 15 and overdispersion 0.2. The second example, using `corrTG()`, computes the Fréchet-Hoeffding upper bounds for the correlations between the gamma random variable with shape 0.5 and rate 1, and gamma random variables with shapes varying from 0.01 to 15 and rate 1. These bounds are plotted in Figure 2, which show that in some scenarios the Fréchet-Hoeffding upper bound can be much lower than 1.

```
R> NBmu <- seq(0.01, 15, by = 0.02)
R> fhub1 <- vector("numeric", length(NBmu))
```

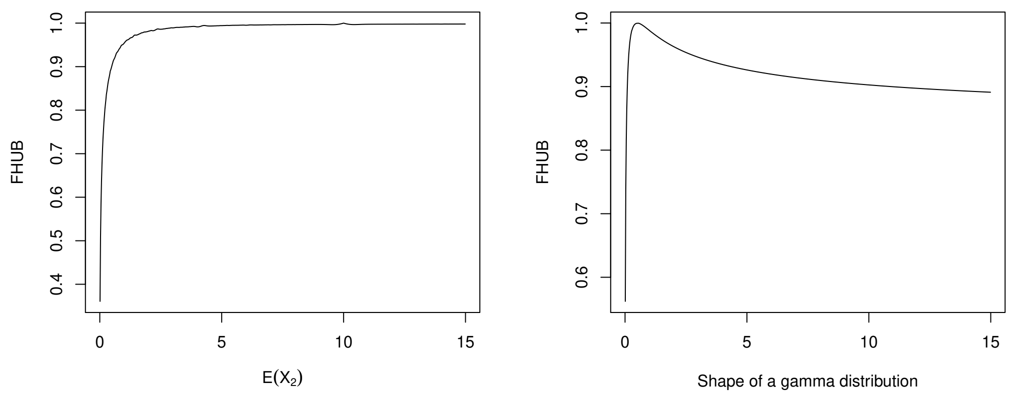


Figure 2: Left: Fréchet-Hoeffding upper bounds between negative binomial random variables. Right: Fréchet-Hoeffding upper bounds between gamma random variables.

```
R> for (i in 1:length(NBmu)) {
+   fhub1[i] <- FHUBdiscrete(marg1 = "nb", marg2 = "nb",
+     mu1 = 10, od1 = 0.2, mu2 = NBmu[i], od2 = 0.2)
+ }
R> gammaShape <- seq(0.01, 15, by = 0.02)
R> fhub2 <- vector("numeric", length(gammaShape))
R> for (i in 1:length(gammaShape)) {
+   fhub2[i] <- corrTG(marg1 = gm.gc(shape = gammaShape[i], rate = 1),
+     marg2 = gm.gc(shape = 0.5, rate = 1), corrGauss = 1,
+     method = "integral")
+ }
R> plot(NBmu, fhub1, type = "l", xlab = expression(E(X[2])), ylab = "FHUB")
R> plot(gammaShape, fhub2, type = "l",
+   xlab = "Shape of a gamma distribution", ylab = "FHUB")
```

3.7. Integrated datasets

The package **gcKrig** includes the following real-world datasets, which have been previously analyzed in the literature:

`AtlanticFish`**:** This dataset contains fish counts sampled at 119 locations in a mid-Atlantic region of the USA, together with several stream characteristics (covariates) related to water quality; the covariates are standardized to have mean 0 and variance 1. It was analyzed by Johnson and Hoeting (2011) to investigate the relation between abundance of pollution tolerant fish and several environmental factors.

`LansingTrees`**:** This dataset was obtained by aggregating the `lansing` dataset in the R package **spatstat** (Baddeley, Turner, and Rubak 2018), which comes from an investigation of a 924 ft × 924 ft (19.6 acres) area in Lansing Woods, Michigan, USA. The original point process data describe the locations of 2,251 trees and their botanical classification (into maples, hickories, black oaks, red oaks, white oaks and miscellaneous trees). The original plot size has been rescaled to the unit square and the number of different types of trees has been counted within squares of length 1/16. See Kazianka (2013a) and Han and De Oliveira (2019) for analyses of this dataset.

`OilWell`: This is a binary dataset that records the locations of successful and unsuccessful drilling oil wells in the northwest shelf of the Delaware basin in New Mexico, USA. This region is densely drilled in some parts, but has also some sparsely drilled subareas. The original dataset was transformed to a central area of about 65 square kilometers. See Hohn (1999, Chapter 6).

`Weed95`: This dataset consists of weed counts and the percentages of organic matter at 270 sampling locations in an agricultural field in Denmark. The weed species *Viola Arvensis* was counted within circular frames of 0.25 square meters each, except for 10 missing sites in the first row. See Christensen and Waagepetersen (2002) for an analysis of this dataset.

4. Examples

In this section we describe brief analyses of the `AtlanticFish`, `Weed95` and `OilWell` datasets to illustrate the fitting of spatial Gaussian copula models using **gcKrig**; Han and De Oliveira (2019) provide an analysis of the `LansingTrees` dataset. All the analyses were run on a 2015 MacBook Pro with 2.8 GHz Intel Core i7 processor, with 8 threads available for parallel processing.

4.1. The `AtlanticFish` data

The `AtlanticFish` dataset was analyzed by Johnson and Hoeting (2011) with the goal of identifying important covariates that affect fish abundance and estimate their effects. They fitted a spatial generalized linear mixed model using a Bayesian approach. Out of nine potential environmental covariates, three stood out as having a significant influence on fish abundance. The important covariates and their respective posterior inclusion probabilities were the Strahler stream order (`ORDER`, 0.883), the percentage of watershed classified as disturbed by human activity (`DISTOT`, 0.787) and the index of fish habitat quality at the stream site (`HAB`, 0.738). The covariate with the fourth highest posterior inclusion probability was the watershed area (`WSA`, 0.426). However, no strong evidence was found for a significant effect of `WSA` on fish abundance.

We fit a spatial Gaussian copula model to this dataset using **gcKrig**. We restrict attention to the aforementioned four environmental covariates, and seek to assess their impacts on fish abundance. The model assumes a family of negative binomial marginals with the log link function, and an isotropic exponential correlation model with a nugget. We fit two models, one with three covariates (`ORDER`, `DISTOT`, `HAB`), and the other with four covariates (`ORDER`, `DISTOT`, `HAB`, `WSA`).

```
R> data("AtlanticFish", package = "gcKrig")
R> Fitfish <- mlegc(y = AtlanticFish[, 3], x = AtlanticFish[, 4:6],
+    locs = AtlanticFish[, 1:2], longlat = TRUE,
+    marginal = negbin.gc(link = "log"),
+    corr = matern.gc(kappa = 0.5, nugget = TRUE), method = "GHK")
R> summary(Fitfish)

Call:
mlegc(y = AtlanticFish[, 3], x = AtlanticFish[, 4:6], locs = AtlanticFish[,
1:2], marginal = negbin.gc(link = "log"), corr = matern.gc(kappa = 0.5,
nugget = TRUE), longlat = TRUE, method = "GHK")

Coefficients of the model:
                Estimate  Std.Error z value  Pr(>|z|)
Intercept         2.3524     0.2198  10.701   < 2e-16 ***
ORDER             0.3808     0.1894   2.010  0.044395 *
DISTOT           -0.6360     0.2050  -3.103  0.001918 **
HAB               0.4952     0.2224   2.227  0.025948 *
overdispersion    3.6210     0.6125   5.912  3.37e-09 ***
range            38.5986    20.8584   1.851  0.064241 .
nugget            0.7072     0.1968   3.595  0.000325 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

log likelihood = -362.02,  AIC = 738.04,  BIC = 757.49,  AICc = 739.04
```

All the regression parameters are significant at level 0.05. As in Johnson and Hoeting (2011), it is found that `ORDER` and `HAB` are positively associated with fish abundance, while `DISTOT` is negatively associated. In addition, there is strong evidence of overdispersion in the response variable, and the large estimate for the nugget parameter together with a small estimate for the range parameter (the median distance between sampling locations is 267.39 km) indicate that the spatial association is modest. The above analysis took about 23 seconds to run.

After the model is fitted, the 95% confidence intervals for the regression and overdispersion parameters were computed with

```
R> profile(Fitfish, par.index = 1, method = "GHK")
```

where the argument `par.index` is varied from 1 to 5. The Wald-type and profile likelihood-based confidence intervals are displayed in Table 2. The two types of confidence intervals are similar, but the Wald-type confidence intervals are narrower.

We also fitted a Gaussian copula model to this dataset, but now adding the environmental covariate `WSA` to the other three, with the code and summary results displayed below.

```
R> Fitfish2 <- mlegc(y = AtlanticFish[, 3], x = AtlanticFish[, 4:7],
+    locs = AtlanticFish[, 1:2], longlat = TRUE,
+    marginal = negbin.gc(link = "log"),
+    corr = matern.gc(kappa = 0.5, nugget = TRUE), method = "GHK")
R> summary(Fitfish2)
```

Coefficient	Wald-type 95% CI	Profile likelihood based 95% CI
Intercept	(1.922, 2.783)	(1.906, 2.880)
`ORDER`	(0.010, 0.752)	(−0.003, 0.753)
`DISTOT`	(−1.038, −0.234)	(−1.023, −0.204)
`HAB`	(0.059, 0.931)	(0.057, 0.941)
Overdispersion	(2.421, 4.821)	(2.632, 5.258)

Table 2: Two types of confidence intervals for marginal parameters in the `AtlanticFish` dataset.

```
Call:
mlegc(y = AtlanticFish[, 3], x = AtlanticFish[, 4:7], locs = AtlanticFish[,
1:2], marginal = negbin.gc(link = "log"), corr = matern.gc(kappa = 0.5,
nugget = TRUE), longlat = TRUE, method = "GHK")

Coefficients of the model:
                Estimate  Std.Error z value  Pr(>|z|)
Intercept        2.35078    0.21893  10.738  < 2e-16 ***
ORDER            0.32454    0.23843   1.361 0.173469
DISTOT          -0.62947    0.20528  -3.066 0.002167 **
HAB              0.49094    0.22207   2.211 0.027055 *
WSA              0.08323    0.22063   0.377 0.705996
overdispersion   3.61206    0.60963   5.925 3.12e-09 ***
range           38.39692   21.01521   1.827 0.067685 .
nugget           0.71073    0.19697   3.608 0.000308 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

log likelihood = -361.94,  AIC = 739.89,  BIC = 762.12,  AICc = 741.19
```

This new fit shows that the effect of `WSA` is not statistically significant. In addition, the model comparisons based on any of the information criteria show that the initial model with the three covariates is preferable. These findings based on a Gaussian copula model are consistent with those reached by Johnson and Hoeting (2011) based on a spatial generalized linear mixed model.

4.2. The `Weed95` data

The `Weed95` dataset was analyzed by Christensen and Waagepetersen (2002) using a generalized linear mixed model, with the goal of investigating the relation between weed occurrence and soil properties. It consists of weed counts and percentage of organic matter at 270 small circular frames within an agricultural field. As the response consists mostly of small counts, generalized linear mixed models may not be able to represent the spatial correlation in these data (De Oliveira 2013). Hence, we fit a Gaussian copula model with negative binomial marginals to the `Weed95` dataset, which has a more flexible correlation structure (Han and De Oliveira 2016).

We illustrate the prediction functionality of the package **gcKrig** using this dataset. Following the analysis in Christensen and Waagepetersen (2002), organic matter and the y (northing) coordinate are used as covariates in the mean model, and an isotropic exponential covariance function with nugget; both covariates are standardized in the range $[-1, 1]$. We use observations of the weed counts at 70 sampling locations to fit the model, and make predictions at the remaining 200 locations. These locations are shown in Figure 3. Rather than predicting weed *intensity* at the latter set of locations, as Christensen and Waagepetersen (2002) did, we predict weed *counts* directly, using the implemented parallel prediction function in `predgc()`. Normally, the maximum number of processing units for parallel computing is equal to the number of threads available in a computer. We suggest to first call the R core package **parallel** and run the following code to detect the number of threads available

```
R> library("parallel")
R> detectCores(all.tests = FALSE, logical = TRUE)
```

and then run the code

```
R> library("snowfall")
R> data("Weed95", package = "gcKrig")
R> weedobs <- subset(Weed95, dummy == 1)
R> weedpred <- subset(Weed95, dummy == 0)
R> predweed <- predgc(obs.y = weedobs$weedcount, obs.x = weedobs[, 4:5],
+    obs.locs = weedobs[, 1:2], pred.x = weedpred[, 4:5],
+    pred.locs = weedpred[, 1:2], marginal = negbin.gc(link = "log"),
+    corr = matern.gc(kappa = 0.5, nugget = TRUE), method = "GHK",
+    pred.interval = 0.95, parallel = TRUE,
+    paralleloptions = list(n.cores = 4, cluster.type = "SOCK"))
R> summary(predweed)
R> plot(predweed, plottype = "2D All", xlab = "Coordinate x (m)",
+    ylab = "Coordinate y (m)", col = c(3, 2), textcex = 0.8)
R> plot(predweed, plottype = "Predicted Variance",
+    xlab = "Coordinate x (m)", ylab = "Coordinate y (m)",
+    col = c(3, 2), textcex = 0.8)
```

The prediction of the counts at the 200 locations using the GHK method with 4 cores took about 39 seconds to run; this includes both the estimation and prediction time. To visually summarize the results, the S3 method `plot()` can be applied to the prediction results with class '`predgc`'. With different choices of the `plottype`, the function can generate five different plots including 2-D level plots for the original data, predicted mean responses and estimated variances at predicted locations. It can also generate a 3-D plot and text plot with the exact counts. Figure 3 displays two of these that map the observations, predictions and the estimated prediction variances.

To assess the accuracy of the predictions, we compute the two types of 95% prediction intervals described in Section 3.4 for all the 200 prediction locations. Out of the 200 locations, the "equal-tail" prediction intervals did not include the actual counts at 11 locations, while the "highest probability mass" predictions intervals did not include the actual counts at 9 locations; so the empirical coverages of these prediction intervals were 94.5% and 95.5%, respectively. The average length of the two types of prediction intervals were 2.815 and 2.445.

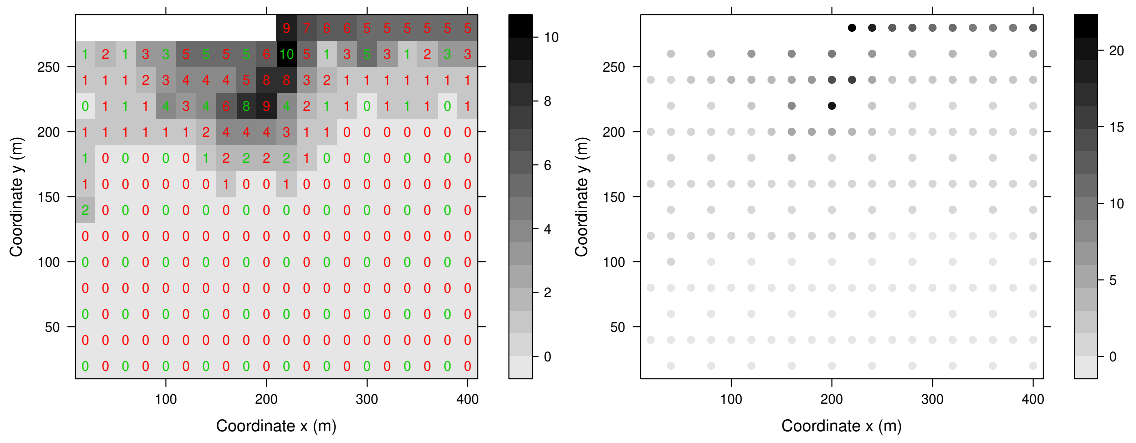


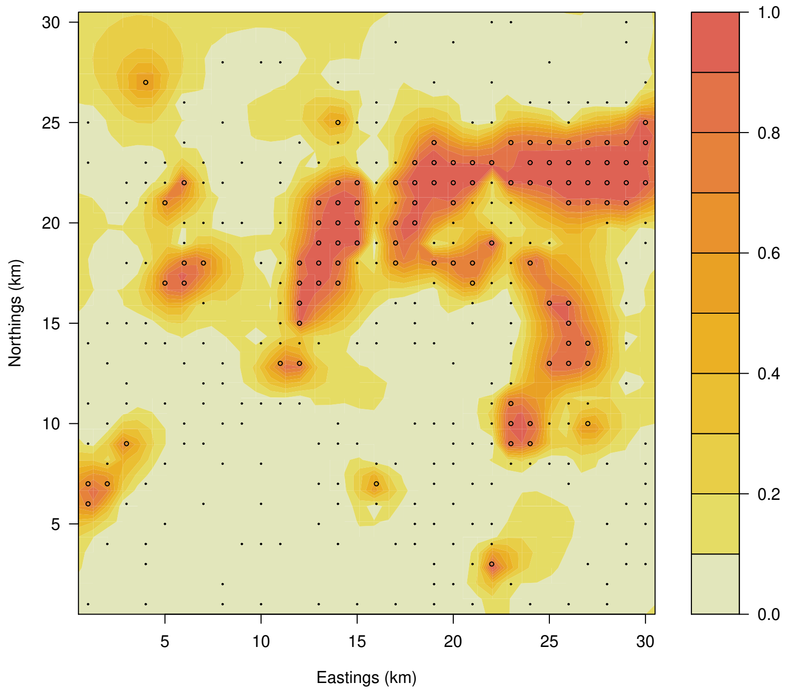
Figure 3: Left: Observed (green) and predicted counts (red). Right: Estimated prediction variances.

These findings indicate that, for this example, the prediction intervals are accurate, with the "highest probability mass" prediction intervals being slightly preferable, as they are on average shorter.

4.3. The `OilWell` data

The `OilWell` dataset was analyzed by Hohn (1999) with the goal of assessing oil abundance in a field in the northwest shelf of the Delaware basin in New Mexico, USA. Oil wells were drilled at 333 locations, with oil found at 100 locations (coded as 1), and oil not found at the remaining locations (coded as 0). The result is a spatial binary dataset in which there were no covariates. The original region with irregular borderlines was rescaled to a central area of about 65 square kilometers. The goal of the analysis is to estimate the probability of oil presence at many unsampled locations. This will help engineers decide where to drill in the future to assess the economic potential of the oil field. Preliminary analyses and information criteria suggest that the fitted model lacks a nugget effect and that an exponential correlation function fits this dataset better than the spherical correlation function and other correlation functions from the Matérn family. This is consistent with the analysis in Hohn (1999) using indicator kriging. We compute the predictive distribution of the response at each location on a 40 $\times$ 40 grid covering the study area. Due to the binary nature of the response, the predictive distribution of $Y(\mathbf{s}_0)$ is determined by a single number, the conditional probability of oil occurrence given the data. This is computed using **gcKrig** with parallel computing using both the GHK and GQT methods. The code below computes both the probability of oil occurrence at 1600 locations and their corresponding prediction variances using the GHK method. It also computes the top graphs in Figures 4 and 5. Replacing the argument `method = "GHK"` with `method = "GQT"` will generate the bottom graphs in Figures 4 and 5.

```
R> data("OilWell", package = "gcKrig")
R> gridstep <- seq(0.5, 30.5, length = 40)
R> locOilpred <- data.frame(Easting = expand.grid(gridstep, gridstep)[, 1],
+    Northing = expand.grid(gridstep, gridstep)[, 2])
```



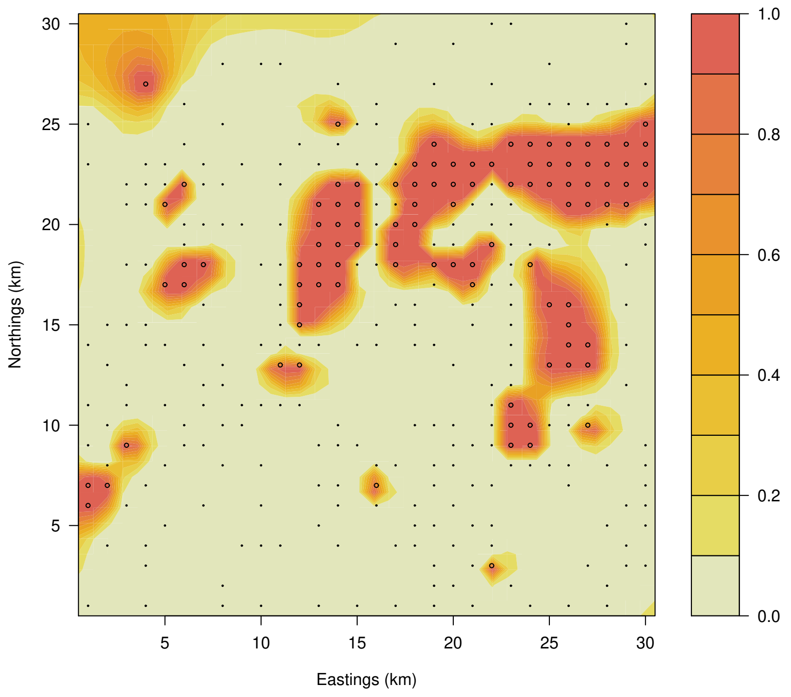
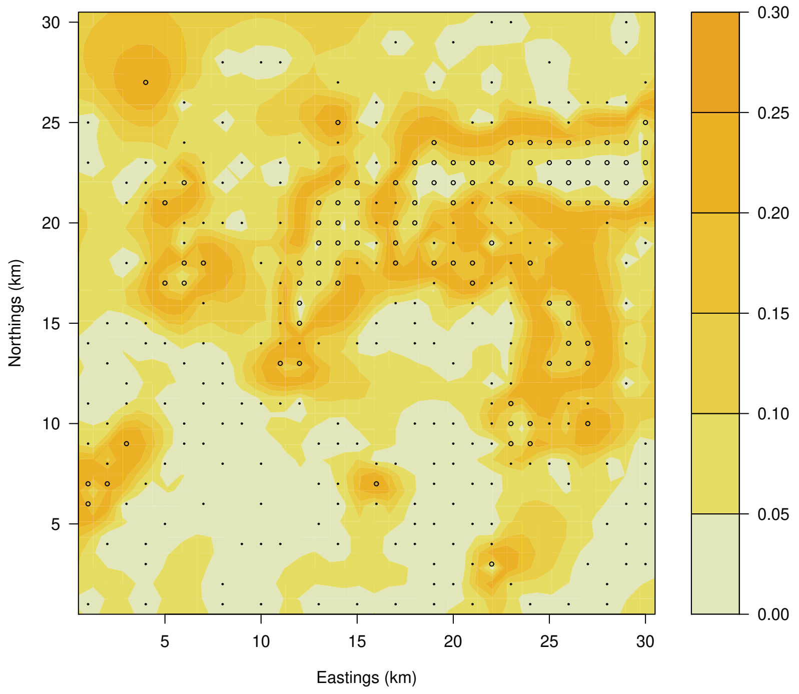


Figure 4: Probability of abundance of crude oil predictive map using GHK method (top) and GQT method (bottom) based on 333 existing drills. Circles represent successful drills and dots represent unsuccessful drills.



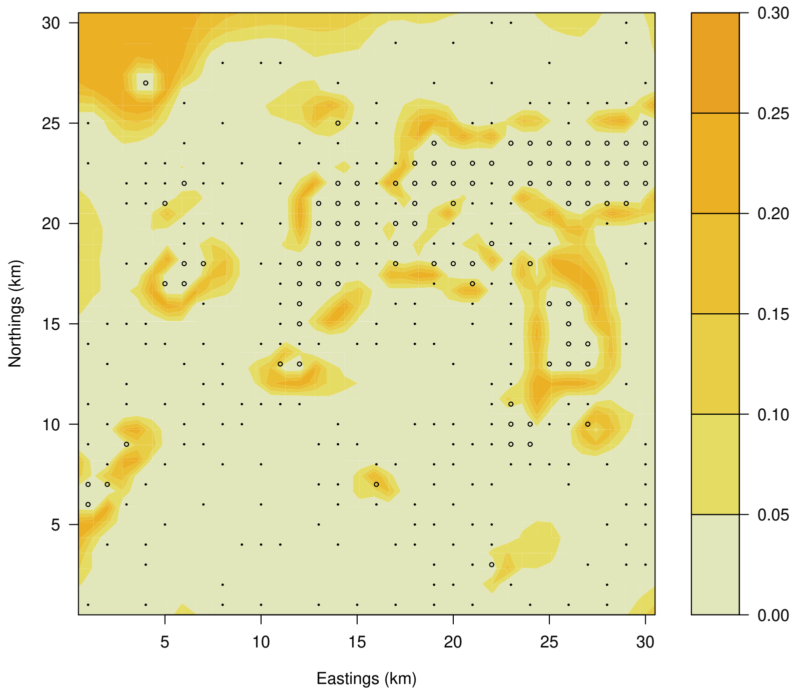


Figure 5: Map of uncertainty about the predictive probabilities using GHK method (top) and GQT method (bottom) based on 333 existing drills. Circles represent successful drills and dots represent unsuccessful drills.

```
R> PredOil <- predgc(obs.y = OilWell[, 3], obs.locs = OilWell[, 1:2],
+    pred.locs = locOilpred, marginal = binomial.gc(link = "logit"),
+    corr = matern.gc(nugget = FALSE), obs.effort = 1,
+    pred.effort = 1, method = "GHK",
+    parallel = TRUE, paralleloptions = list(n.cores = 4))
R> PredMat <- summary(PredOil)
R> library("colorspace")
R> filled.contour(seq(0.5, 30.5, length = 40),
+    seq(0.5, 30.5, length = 40), matrix(PredMat$predMean, 40),
+    zlim = c(0, 1), col = rev(heat_hcl(12)), nlevels = 12,
+    xlab = "Eastings (km)", ylab = "Northings (km)",
+    plot.axes = {axis(1); axis(2); points(OilWell[, 1:2], col = 1,
+      cex = 0.2 + 0.4 * OilWell[, 3])})
R> filled.contour(seq(0.5, 30.5, length = 40),
+    seq(0.5, 30.5, length = 40), matrix(PredMat$predVar, 40),
+    zlim = c(0, 0.3), col = rev(heat_hcl(12)), nlevels = 10,
+    xlab = "Eastings (km)", ylab = "Northings (km)",
+    plot.axes = {axis(1); axis(2); points(OilWell[, 1:2],
+      col = 1, cex = 0.2 + 0.4 * OilWell[, 3])})
```

The computation of the above predictive distributions at 1600 locations based on 4 cores took about 556 seconds for the GHK method and 172 seconds for the GQT method; these include both the estimation and prediction time. In general terms both methods provide similar results. It is more likely to have a successful drill in the areas where the majority of the existing drills were successful, and it is unlikely to have a successful drill in the areas surrounded by dry wells. Also, the predictive variances are smaller in the places where the prediction locations are closer to sampling locations, especially in the areas containing mostly only one type of well (successful or unsuccessful). But there are two main differences between the maps obtained by these methods. The map of the probability of oil occurrence obtained from the GHK method is smoother than that obtained from the GQT method, and the predictive uncertainty measures from the GQT method are smaller; the latter are likely to underestimate the true prediction uncertainty. This illustrates the inaccuracy of the GQT method, alluded to in Sections 2.1 and 2.2, when the "discreteness" in the data is extreme.

5. Comparison with other packages

In this section we provide a comparison between **gcKrig** and two other R packages, **mvtnorm** and **gcmr**. The package **mvtnorm** (Genz *et al.* 2018) approximates general n-dimensional integrals of normal (and t) densities, Equation 7 being a special case, using a quasi-Monte Carlo method; Nikoloulopoulos (2013) used it to make inference about Gaussian copula models with discrete marginals. The package **gcmr** (Masarotto and Varin 2017, 2018) can fit Gaussian copula models to longitudinal, time series and spatial data, with both continuous and discrete marginals. It uses the GHK method when fitting the models to discrete data. The packages are compared by fitting a simulated dataset over the region $[0, 1] \times [0, 1]$, with 121 sampling locations forming a regular 11×11 grid. The model for the simulated data has negative binomial marginals with mean function $\exp(1 + 0.5x + y)$, $\mathbf{s} = (x, y)$, overdispersion 1 and correlation function of the latent field $\exp(-\|\mathbf{s} - \mathbf{u}\|/0.3)$ (no nugget).

```
R> xloc <- rep(seq(0, 1, by = 0.1), 11)
R> yloc <- rep(seq(0, 1, by = 0.1), each = 11)
R> simD <- as.matrix(dist(cbind(xloc, yloc)))
R> set.seed(321)
R> simData1 <- simgc(locs = cbind(xloc, yloc), sim.n = 1,
+    marginal = negbin.gc(mu = exp(1 + 0.5 * xloc + yloc), od = 1),
+    corr = matern.gc(range = 0.3, nugget = 0))
R> simDf1 <- data.frame(xloc = xloc, yloc = yloc, data = simData1$data)
```

Since **mvtnorm** is not specifically designed to make inference about Gaussian copula models, the function below calls **mvtnorm** for the evaluation of the log-likelihood of the parameters of the aforementioned model.

```
R> QMCLik <- function(v) {
+    mu <- exp(v[1] + v[2] * xloc + v[3] * yloc)
+    lower <- qnorm(pnbinom(q = simDf1$data - 1, size = 1/v[4], mu = mu))
+    upper <- qnorm(pnbinom(q = simDf1$data, size = 1/v[4], mu = mu))
+    R <- exp(-simD/v[5])
+    set.seed(1234)
+    lik <- -log(mvtnorm::pmvnorm(lower = lower, upper = upper,
+      mean = rep(0, length(lower)), sigma = R)[1])
+    return(lik)
+  }
```

The code below fits the aforementioned Gaussian copula model to the simulated data `simDf1` using **mvtnorm**, **gcmr** and **gcKrig**. For the latter, both GHK and GQT methods are used. The same initial values are used in all fits.

```
R> est1 <- MASS::glm.nb(data ~ xloc + yloc, data = simDf1)
R> start0 <- c(coef(est1), 1/est1$theta, median(simD)/2)
R> EPS <- .Machine$double.eps
R> library("mvtnorm")
R> library("numDeriv")
R> t0 <- proc.time()
R> GBFit <- optim(par = start0, fn = QMCLik, gr = NULL,
+    method = c("L-BFGS-B"), lower = c(-Inf, -Inf, -Inf, EPS, EPS),
+    upper = c(Inf, Inf, Inf, Inf, Inf))
R> tGB <- proc.time() - t0
R> HessGB <- hessian(QMCLik, x = GBFit$par)
R> sdGB <- sqrt(diag(solve(HessGB)))
R> library("gcmr")
R> t0 <- proc.time()
R> gcmrGHK <- gcmr(data ~ xloc + yloc, data = simDf1,
+    marginal = negbin.marg(link = "log"),
+    cormat = matern.cormat(D = simD),
+    options = gcmr.options(seed = 123, nrep = c(100, 1000)))
R> tgcmr <- proc.time() - t0
```

Parameter	**gcmr**	**gcKrig**-GHK	**gcKrig**-GQT	**mvtnorm**
β_0	1.423 (0.463)	1.442 (0.458)	1.434 (0.467)	1.422 (0.590)
β_1	0.553 (0.511)	0.538 (0.506)	0.519 (0.521)	0.553 (0.561)
β_2	0.672 (0.521)	0.665 (0.518)	0.656 (0.540)	0.673 (0.573)
σ^2	0.550 (0.314)	0.544 (0.297)	0.615 (0.350)	0.551 (0.370)
θ	0.205 (0.094)	0.203 (0.090)	0.227 (0.107)	0.206 (0.120)
Log-likelihood	−357.60	−357.60	−357.37	−357.56
Time (seconds)	15.7	13.0	4.5	431.0

Table 3: Parameter estimates with estimated standard deviations in parenthesis and the log-likelihoods evaluated at the parameter estimates using different methods/packages, as well as the corresponding computational times.

```
R> library("gcKrig")
R> t0 <- proc.time()
R> gcKrigGHK <- mlegc(y = simDf1[, 3], x = simDf1[, 1:2],
+    locs = cbind(xloc,yloc), method = "GHK",
+    marginal = negbin.gc(link = "log"),
+    corr = matern.gc(nugget = FALSE),
+    ghkoptions = list(nrep = c(100, 1000), seed = 123))
R> tgcKrigGHK <- proc.time() - t0
R> t0 <- proc.time()
R> gcKrigGQT <- mlegc(y = simDf1[, 3], x = simDf1[, 1:2],
+    locs = cbind(xloc,yloc), method = "GQT",
+    marginal = negbin.gc(link = "log"),
+    corr = matern.gc(nugget = FALSE))
R> tgcKrigGQT <- proc.time() - t0
```

Table 3 collects the results. Both the estimates and their standard errors (in parentheses) are quite similar for all the packages. On the other hand, the running time of **mvtnorm** is substantially larger than those of the others. The running times of **gcmr** and **gcKrig** with the GHK method are about the same, while **gcKrig** with the GQT method runs in less than half of the time. Then, the latter is a useful option when fitting moderately large datasets.

When comparing **gcmr** and **gcKrig** in terms of other capabilities, we note that **gcKrig** can perform predictive inference at unobserved locations, as well as computation of the correlation function of the random field of counts and Fréchet-Hoeffding upper bounds, while these tasks are not available in **gcmr**. On the other hand, **gcmr** computes a type of residuals relevant for assessing the fit of Gaussian copula models, while **gcKrig** does not (currently) perform this task.

Finally, we note that the current version of **gcmr** does not seem to handle adequately correlation functions that include a nugget effect. Although the currently available options for the correlation structure in **gcmr** do not include a nugget effect in the correlation matrix, the user can write a function of class '`cormat.gcmr`' that includes a nugget effect; see Masarotto and Varin (2017). But when we explored this for several simulated datasets, the estimated nugget parameter turned out to be negative for some datasets, mostly when the true nugget parameter was close to zero. This is presumably due to the optimization algorithm used in **gcmr** not constraining the nugget parameter to be in $[0, 1]$.

6. Conclusions

In this work we described the use of the R package **gcKrig** for the analysis of geostatistical count data using Gaussian copulas. The package implements most of the main tasks commonly required in the analysis of this kind of data: simulation and visualization, parameter (point and interval) estimation, spatial (point and interval) prediction, and computation of the correlation function of the process of counts. The inferential tasks rely either on an accurate approximation to the likelihood (the GHK simulator) or on a surrogate likelihood (the GQT method); the latter is appealing for the analysis of moderately large datasets. The package implements the computationally intensive tasks in C++ using an R/C++ interface, and has parallel computing capabilities to predict the response at multiple locations simultaneously. Nevertheless, the more accurate method based on the GHK simulator cannot handle large datasets effectively. Hence, we plan to enhance the package in the future with the implementation of methods to base inference on pairwise likelihoods or other efficient surrogate likelihoods.

Another possible modeling approach is the use of copulas based on graphical models called *vines*, along the lines described in Panagiotelis, Czado, and Joe (2012). Gräler (2014) and Erhardt, Czado, and Schepsmeier (2015a,b) used this approach for the modeling of geostatistical continuous data, which is implemented in the R packages **spcopula** (Gräler 2017) and **VineCopula** (Schepsmeier, Stoeber, Brechmann, Graeler, Nagler, and Erhardt 2018). The application of this approach to the modeling of geostatistical count data seems a promising topic of future research.

Acknowledgments

We thank two anonymous referees for their comments and suggestions, specially for bringing to our attention recent work on spatial vine copula models. This work was partially supported by the US National Science Foundation Grant DMS-1208896.

References

Baddeley A, Turner R, Rubak E (2018). ***spatstat**: Spatial Point Pattern Analysis, Model-Fitting, Simulation, Tests.* R package version 1.56-1, URL `https://CRAN.R-project.org/package=spatstat`.

Bai Y, Kang J, Song PXK (2014). "Efficient Pairwise Composite Likelihood Estimation for Spatial-Clustered Data." *Biometrics*, **70**(3), 661–670. `doi:10.1111/biom.12199`.

Botev Z (2017). "The Normal Law Under Linear Restrictions: Simulation and Estimation Via Minimax Tilting." *Journal of the Royal Statistical Society B*, **79**(1), 125–148. `doi:10.1111/rssb.12162`.

Cameron AC, Trivedi PK (2013). *Regression Analysis of Count Data.* 2nd edition. Cambridge University Press, Cambridge. `doi:10.1017/CBO9781139013567`.

Christensen OF, Waagepetersen R (2002). "Bayesian Prediction of Spatial Count Data Using Generalized Linear Mixed Models." *Biometrics*, **58**(2), 280–286. doi:10.1111/j.0006-341x.2002.00280.x.

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

De Oliveira V (2003). "A Note on the Correlation Structure of Transformed Gaussian Random Fields." *Australian and New Zealand Journal of Statistics*, **45**(3), 353–366. doi:10.1111/1467-842x.00289.

De Oliveira V (2013). "Hierarchical Poisson Models for Spatial Count Data." *Journal of Multivariate Analysis*, **122**, 393–408. doi:10.1016/j.jmva.2013.08.015.

Demirtas H, Hedeker D (2011). "A Practical Way for Computing Approximate Lower and Upper Correlation Bounds." *The American Statistician*, **65**(2), 104–109. doi:10.1198/tast.2011.10090.

Diggle PJ, Tawn JA, Moyeed RA (1998). "Model-Based Geostatistics." *Journal of the Royal Statistical Society C*, **47**(3), 299–326. doi:10.1111/1467-9876.00113.

Erhardt TM, Czado C, Schepsmeier U (2015a). "R-Vine Models for Spatial Time Series with an Application to Daily Mean Temperature." *Biometrics*, **71**(2), 323–332. doi:10.1111/biom.12279.

Erhardt TM, Czado C, Schepsmeier U (2015b). "Spatial Composite Likelihood Inference Using Local C-Vines." *Journal of Multivariate Analysis*, **138**, 74–88. doi:10.1016/j.jmva.2015.01.021.

Genest C, Nešlehová J (2007). "A Primer on Copulas for Count Data." *ASTIN Bulletin: The Journal of the IAA*, **37**(2), 475–515. doi:10.1017/s0515036100014963.

Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*. Springer-Verlag, Heidelberg.

Genz A, Bretz F, Miwa T, Mi X, Hothorn T (2018). ***mvtnorm**: Multivariate Normal and t Distributions*. R package version 1.0-8, URL https://CRAN.R-project.org/package=mvtnorm.

Geweke J (1991). "Efficient Simulation From the Multivariate Normal and Student-*t* Distributions Subject to Linear Constraints." In *Computer Science and Statistics: Proceedings of the Twenty Third Symposium on the Interface*, pp. 571–578. Fairfax Station.

Gibson GJ, Glasbey CA, Elston DA (1994). "Monte Carlo Evalution of Multivariate Normal Integrals and Sensitivity to Variate Ordering." In *Advances in Numerical Methods and Applications: Proceedings of the Third International Conference*, pp. 120–126.

Gräler B (2014). "Modelling Skewed Spatial Random Fields through the Spatial Vine Copula." *Spatial Statistics*, **10**, 87–102. doi:10.1016/j.spasta.2014.01.001.

Gräler B (2017). ***spcopula**: Copula Driven Analysis – Multivariate, Spatial, Spatio-Temporal*. R package version 0.2-4, URL http://R-forge.R-project.org/projects/spcopula/.

Grigoriu M (2007). "Multivariate Distributions with Specified Marginals: Applications to Wind Engineering." *Journal of Engineering Mechanics*, **133**(2), 174–184. doi:10.1061/(asce)0733-9399(2007)133:2(174).

Hajivassiliou V, McFadden D, Ruud P (1996). "Simulation of Multivariate Normal Rectangle Probabilities and Their Derivatives: Theoretical and Computational Results." *Journal of Econometrics*, **72**(1–2), 85–134. doi:10.1016/0304-4076(94)01716-6.

Han Z (2018). ***gcKrig**: Analysis of Geostatistical Count Data Using Gaussian Copulas.* R package version 1.1.3, URL https://CRAN.R-project.org/package=gcKrig.

Han Z, De Oliveira V (2016). "On the Correlation Structure of Gaussian Copula Models for Geostatistical Count Data." *Australian and New Zealand Journal of Statistics*, **58**(1), 47–69. doi:10.1111/anzs.12140.

Han Z, De Oliveira V (2019). "Maximum Likelihood Estimation of Gaussian Copula Models for Geostatistical Count Data." *Communications in Statistics – Simulation and Computation*. Forthcoming.

Hohn M (1999). *Geostatistics and Petroleum Geology*. 2nd edition. Kluwer Academic Publishers, Dordrecht.

Jeske DR (1993). "Predicting the Value of an Integer-Valued Random Variable." *Statistics and Probability Letters*, **16**(4), 297–300. doi:10.1016/0167-7152(93)90133-4.

Johnson DS, Hoeting JA (2011). "Bayesian Multimodel Inference for Geostatistical Regression Models." *PLOS ONE*, **6**(11). doi:10.1371/journal.pone.0025677. E25677.

Kang J, Bai Y, Song PXK (2014). ***GeoCopula**: Efficient Pairwise Composite Likelihood Estimation for Spatial-Clustered Data.* R package version 1.0, URL http://www-personal.umich.edu/~jiankang/software/GeoCopula.html.

Kazianka H (2013a). "Approximate Copula-Based Estimation and Prediction of Discrete Spatial Data." *Stochastic Environmental Research and Risk Assessment*, **27**(8), 2015–2026. doi:10.1007/s00477-013-0737-7.

Kazianka H (2013b). "**spatialCopula**: A MATLAB Toolbox for Copula-Based Spatial Analysis." *Stochastic Environmental Research and Risk Assessment*, **27**(1), 121–135. doi:10.1007/s00477-012-0571-3.

Kazianka H, Pilz J (2010). "Copula-Based Geostatistical Modeling of Continuous and Discrete Data Including Covariates." *Stochastic Environmental Research and Risk Assessment*, **24**(5), 661–673. doi:10.1007/s00477-009-0353-8.

Keane MP (1994). "A Computationally Practical Simulation Estimator for Panel Data." *Econometrica*, **62**(1), 95–116. doi:10.2307/2951477.

Knaus J (2015). ***snowfall**: Easier Cluster Computing (based on **snow**.* R package version 1.84-6.1, URL https://CRAN.R-project.org/package=snowfall.

Madsen L (2009). "Maximum Likelihood Estimation of Regression Parameters with Spatially Dependent Discrete Data." *Journal of Agricultural, Biological, and Environmental Statistics*, **14**(4), 375–391. doi:10.1198/jabes.2009.07116.

Masarotto G, Varin C (2012). "Gaussian Copula Marginal Regression." *Electronic Journal of Statistics*, **6**, 1517–1549. doi:10.1214/12-ejs721.

Masarotto G, Varin C (2017). "Gaussian Copula Regression in R." *Journal of Statistical Software*, **77**(8), 1–26. doi:10.18637/jss.v077.i08.

Masarotto G, Varin C (2018). ***gcmr**: Gaussian Copula Marginal Regression.* R package version 1.0.1, URL https://CRAN.R-project.org/package=gcmr.

Meeker WQ, Escobar LA (1995). "Teaching About Approximate Confidence Regions Based on Maximum Likelihood Estimation." *The American Statistician*, **49**(1), 48–53. doi:10.1080/00031305.1995.10476112.

Nelsen RB (1987). "Discrete Bivariate Distributions with Given Marginals and Correlation." *Communications in Statistics – Simulation and Computation*, **16**(1), 199–208. doi:10.1080/03610918708812585.

Nelsen RB (2006). *An Introduction to Copulas.* 2nd edition. Springer-Verlag, New York. doi:10.1007/0-387-28678-0.

Nikoloulopoulos AK (2013). "On the Estimation of Normal Copula Discrete Regression Models Using the Continuous Extension and Simulated Likelihood." *Journal of Statistical Planning and Inference*, **143**(11), 1923–1937. doi:10.1016/j.jspi.2013.06.015.

Nikoloulopoulos AK (2016). "Efficient Estimation of High-Dimensional Multivariate Normal Copula Models with Discrete Spatial Responses." *Stochastic Environmental Research and Risk Assessment*, **30**(2), 493–505. doi:10.1007/s00477-015-1060-2.

Panagiotelis A, Czado C, Joe H (2012). "Pair Copula Constructions for Multivariate Discrete Data." *Journal of the American Statistical Association*, **107**(499), 1063–1072. doi:10.1080/01621459.2012.682850.

R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ridgway J (2016). "Computation of Gaussian Orthant Probabilities in High Dimension." *Statistics and Computing*, **26**(4), 899–916. doi:10.1007/s11222-015-9578-1.

Schepsmeier U, Stoeber J, Brechmann EC, Graeler B, Nagler T, Erhardt T (2018). ***VineCopula**: Statistical Analysis of Vine Copulas.* R package version 2.1.8, URL https://CRAN.R-project.org/package=VineCopula.

The MathWorks Inc (2017). *MATLAB – The Language of Technical Computing, Version R2017b.* Natick. URL http://www.mathworks.com/products/matlab/.

Varin C, Reid N, Firth D (2011). "An Overview of Composite Likelihood Methods." *Statistica Sinica*, **21**(1), 5–42.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S.* New York, 4th edition. doi:10.1007/978-0-387-21706-2.

A. Specification prototypes

This appendix provides prototypes for users to specify marginals, link functions and correlation functions. We first show how to construct a marginal of class '`marginal.gc`' with a user-defined link function. Then, we provide source code as an example on how to specify correlation functions of class '`corr.gc`'. These prototypes serve as templates and can be easily modified by interested users to construct user-defined marginals, link functions and correlation functions that are not yet available in the package **gcKrig**.

A.1. Specification of a new marginal of class '`marginal.gc`'

This example provides the source code for specifying the binomial distribution with link function t and `df.t` degrees of freedom as marginal of class '`marginal.gc`'. This function can be an input for the argument `marginal` in the functions `simgc()`, `corrTG()`, `mlegc()` and `predgc()`, and serves as prototype to construct user-defined marginals and link functions.

```
binomial_t.gc <- function(df.t = 6, size = NULL, prob = NULL)
{
  ans <- list()
  ans$discrete <- TRUE
  if(is.null(size) & is.null(prob)){
    ans$start <- function(y, x, effort) {
      mfit <- suppressWarnings(glm.fit(x, y/effort,
                                  family = binomial(link = "logit")))
      reg0 <- coef(mfit)
      glb <- qnorm(pbinom(y - 1, size = effort, prob = fitted(mfit)))
      gub <- qnorm(pbinom(y, size = effort, prob = fitted(mfit)))
      names(reg0)[1] <- "Intercept"
      return(reg0)
    }
    ans$nod <- 0
    ans$bounds <- function(y, x, pars, effort) {
      p <- pt(pars[1:ncol(x)]%*%t(x), df = df.t)
      a <- qnorm(pbinom(y - 1, size = effort, prob = p))
      b <- qnorm(pbinom(y, size = effort, prob = p))
      return(list(lower = a, upper = b))
    }
    ans$pdf <- function(y, x, pars, effort){
      p <- pt(pars[1:ncol(x)]%*%t(x), df = df.t)
      pdf <- dbinom(y, size = effort, prob = p, log = FALSE)
      return(pdf)
    }
    ans$cdf <- function(y, x, pars, effort){
      p <- pt(pars[1:ncol(x)]%*%t(x), df = df.t)
      cdf <- pbinom(y, size = effort, prob = p)
      return(cdf)
    }
  }
  if(is.numeric(size) & is.numeric(prob))
  {
    q <- function(p) qbinom(p = p, size = size, prob = prob)
    ans$margvar <- size*prob*(1-prob)
    ans$int.marg <- function (order){
      if(requireNamespace("EQL", quietly = TRUE)){
        integrate(function(x, order)
          ifelse((q(pnorm(x))==Inf | dnorm(x) < .Machine$double.eps), 0,
                 q(pnorm(x))*dnorm(x)*EQL::hermite(x, order, prob = TRUE)),
          order = order, -8, 8, subdivisions = 1500,
          rel.tol = 0.01, stop.on.error = FALSE)
      }else{
        stop("Please install {EQL} first!")
      }
    }
    ans$rsample <- function(nrep){
      rbinom(n = nrep, size = size, prob = prob)
    }
    ans$q <- q
  }
  class(ans) <- c("marginal.gc")
  return(ans)
}
```

The output of this function is a list with the following components:

`discrete` indicates whether the marginal is discrete or not. For continuous marginals, the package can only be used for data simulation and non-Gaussian correlation computation. In this case, `start`, `nod`, `bounds`, `pdf` and `cdf` of the list are no longer needed.

`start` is a function of the response vector `y`, the matrix or data frame of the covariates `x` and the sampling effort `effort`. The output of `start()` is a vector of the starting values for marginal parameters.

`nod` indicates whether the marginal has an overdispersion parameter (`nod = 1`) or not (`nod = 0`).

`bounds` is a function of the response vector `y`, the matrix or data frame of the covariates `x`, the vector of parameters `pars` and the sampling efforts `effort`. This function computes the upper and lower limits of the integral in Equation 7.

`pdf` is a function of the response vector `y`, the matrix or data frame of the covariates `x`, the vector of parameters `pars` and the sampling efforts `effort`. This function returns a vector of the marginal probabilities of all observations.

`cdf` is a function of the response vector `y`, the matrix or data frame of the covariates `x`, the vector of parameters `pars` and the sampling efforts `effort`. This function returns a vector of the marginal cumulative distribution function of all observations.

`margvar` returns the variance of the distribution for the given set of parameters.

`int.marg` is a function that returns the coefficients in Equation 15.

`rsample` is a function that generates a random sample of size `nrep` from this distribution.

`q` is the quantile function defined above.

A.2. Specification of a new correlation of class 'corr.gc'

This example provides the source code for specifying the power exponential correlation function of class '`corr.gc`', which can be the input for the argument `corr` in the function `simgc()`, `mlegc()` and `predgc()`. It serves as a prototype to construct different correlation functions for geostatistical data simulation, model fitting and spatial prediction.

```
powerexp.gc <- function (range = NULL, kappa = 1, nugget = TRUE)
{
  ans <- list()
  if (kappa > 2)
    stop("'Kappa' must be between 0 and 2")
  if (is.null(range)) {
    if (nugget == TRUE) {
      ans$nug <- 1
      ans$npar.cor <- 2
      ans$start <- function(D) {
        corstart <- c(median(D)/2, 0.2)
        names(corstart) <- c("range", "nugget")
        return(corstart)
      }
      ans$corr <- function(corrpar, D) {
        S <- (1 - corrpar[2]) * exp(-abs((D/corrpar[1])^(kappa))) +
          corrpar[2] * diag(NROW(D))
        return(S)
      }
    }
    else {
      ans$nug <- 0
      ans$npar.cor <- 1
      ans$start <- function(D) {
        corstart <- median(D)/2
        names(corstart) <- "range"
        return(corstart)
       }
      ans$corr <- function(corrpar, D) {
        S <- exp(-abs((D/corrpar[1])^(kappa)))
        return(S)
      }
    }
  }
  if (is.numeric(range) & is.numeric(nugget)) {
    ans$S <- function(D) (1 - nugget) * exp(-abs((D/range)^(kappa))) +
      nugget * diag(NROW(D))
  }
  class(ans) <- c("corr.gc")
  return(ans)
}
```

The output is a list with five components:

`nug` indicates whether the correlation model has a nugget parameter (`nugget = TRUE`) or not (`nugget = FALSE`).

`npar.cor` indicates the number of correlation parameters to be estimated in the model, except the nugget parameter.

`start` is a function of the distance matrix `D` that returns the starting values of the correlation parameters for optimization.

`corr` is a function of the correlation parameter `corrpar` and the distance matrix `D` that returns the formula of the correlation function. It is used for likelihood inference and spatial prediction.

`S` is a function of the distance matrix `D` only, which specifies the correlation function. It is used for simulation purpose only, in which both `range` and `nugget` are known.

Affiliation:

Zifei Han
Vertex Pharmaceuticals
50 Northern Ave
Boston MA 02210, United States of America
E-mail: hanzifei1@gmail.com

Victor De Oliveira
Department of Management Science and Statistics
The University of Texas at San Antonio
San Antonio TX 78249, United States of America
E-mail: victor.deoliveira@utsa.edu
URL: http://faculty.business.utsa.edu/vdeolive/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
December 2018, Volume 87, Issue 13 *Submitted:* 2017-04-18
doi:10.18637/jss.v087.i13 *Accepted:* 2017-12-03